REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-03-2007	Final Report	March-2004 to Feb-2007

4. TITLE AND SUBTITLE
A Heuristic Design Information Sharing Framework for Hard Discrete Optimization Problems

5a. CONTRACT NUMBER
FA9550-04-1-0110

5b. GRANT NUMBER
6RNM25

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Jacobson, Sheldon H., Ph.D.

5d. PROJECT NUMBER
FQ867100400698

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)

University of Illinois
Department of Computer Science
Thomas M. Siebel Center for Computer Science
201 N Goodwin Ave (MC-258)
Urbana, IL 61801-2302

8. PERFORMING ORGANIZATION REPORT NUMBER

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Fariba Fahroo, Ph.D., Don Hearn, Ph.D.
Program Manager: Optimization & Discrete Mathematics
Air Force Office of Scientific Research
Suite 325, Room 3112
875 Randolph Street, Arlington, VA 22203-1768

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Public Availability

AFRL-SR-AR-TR-07-0143

13. SUPPLEMENTARY NOTES

None

20070516069

14. ABSTRACT

This project studied and developed *simultaneous generalized hill climbing* (SGHC) algorithms as an algorithmic framework for information sharing in discrete optimization problems. This framework has been used to gain new insights into neighborhood structure designs that allow different neighborhood functions to share information when using the same heuristic applied to the same problem. The results reported from this project introduce the SGHC algorithm framework for information sharing across sets of related discrete optimization problems, provide guidelines on how to use and to design neighborhood functions that results in effective performance of local search algorithms, and describe how tabu search can be effectively used to improve the performance of generalized hill climbing algorithms. Extensive computationally results arte reported on a large variety of test bed, large-scale, real-world discrete optimization problems. The primary application for this research were a military combat search and rescue problems, where several possible search and rescue strategies must be considered to determine the optimal strategy, and a homeland security aviation security baggage screening problems, where several different baggage screening strategies at a set of airports must be considered to determine the optimal strategy for the entire system. Both these problems are intractable due, in part, to the exponentially large number of possible solutions that exist and must be evaluated to identify those that are optimal.

15. SUBJECT TERMS
Local Search, Heuristics, Discrete Optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Sheldon H. Jacobson
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU		19b. TELEPHONE NUMBER *(include area code)*
UU	UU	UU			217-244-7275

FINAL TECHNICAL REPORT

Submitted to:

Fariba Fahroo, Ph.D.
Don Hearn, Ph.D.
Program Manager: Optimization and Discrete Mathematics
Directorate of Mathematics and Information Sciences
Air Force Office of Scientific Research
875 North Randolph Street
Arlington, VA 22203

Principal Investigator:

Sheldon H. Jacobson, Ph.D.
Director, Simulation and Optimization Laboratory
Department of Computer Science
University of Illinois
Urbana, IL 61801-2302
(217) 244-7275 (office)
shj@uiuc.edu
netfiles.uiuc.edu/shj/www/shj.html

Grant Number:

FA9550-04-1-0110

TABLE OF CONTENTS

Report Documentation Page 1
Cover Page 2
Table of Contents 3
Executive Summary 4
1. Generalized Hill Climbing Algorithms 5
2. Tabu Guided Generalized Hill Climbing Algorithms 10
3. Simultaneous Generalized Hill Climbing Algorithms 15
4. Neighborhood Function Design Properties 25
5. Threshold Analysis Performance 34
6. Other Research Results 41
References 43
Contributing Personnel 46
Transitions 46
Publications 47
Awards/Honors 48

EXECUTIVE SUMMARY

The research conducted under this grant focused on an extensive mathematical analysis of algorithms within the generalized hill climbing algorithm framework for addressing intractable discrete optimization problems. The major technical accomplishments achieved include

i) a rigorous analysis of generalized hill climbing algorithms that focused on visiting rather than convergence criteria, which more closely matched how such algorithms are used in practice.

ii) an introduction of tabu guided generalized hill climbing algorithms, which shows how tabu search can be used to enhance the performance of generalized hill climbing algorithms

iii) a rigorous analysis of simultaneous generalized hill climbing algorithms that focused on performance criteria and conditions,

iv) a rigorous analysis of neighborhood properties for generalized hill climbing algorithms ,

v) an introduction and analysis of β-acceptable finite-time performance measures for generalized hill climbing algorithms,

Several other problems were studied during the grant period, including a complexity analysis of discrete random number generators based on the alias method, aviation security system design and optimization problems, and a pediatric vaccine formulary design problem.

All the accomplishments resulting form the research reported under this grant are documented in several archival journal articles and conference proceedings. Many of the results have also been presented at national and international conferences, and have won awards for their contribution. Several of the results of this research have been transitioned to industrial and government agencies, including Austral Engineering and Software Inc., and the Homeland Security Institute within the United States Department of Homeland Security.

Three Ph.D. dissertations were completed during the period of the grant. Dr. Laura Ann McLay successfully defended and submitted her Ph.D. dissertation "Designing Aviation Security Systems: Theory and Practice" in May 2006. Dr. Hemanshu Kaul successfully defended and submitted his Ph.D. dissertation "Topics in Stochastic Combinatorial Optimization and Extremal Graph Theory" in August 2006. Major Shane N. Hall, USAF, successfully defended and submitted his Ph.D. dissertation " The Design and Analysis of Pediatric Vaccine Formularies: Theory and Practice " in August 2006.

1. Generalized Hill Climbing Algorithms

Generalized hill climbing algorithms provide a framework for modeling several local search algorithms for hard discrete optimization problems. Jacobson and Yucesan (2004a) introduce and analyze generalized hill climbing algorithm performance measures that reflect how effectively an algorithm has performed to date in visiting a global optimum and how effectively an algorithm may perform in the future in visiting such a solution. These measures are also used to obtain a necessary asymptotic convergence (in probability) condition to a global optimum, which is then used to show that a common formulation of threshold accepting does not converge. These measures assume particularly simple forms when applied to specific search strategies such as Monte Carlo search and threshold accepting.

To describe these results, the following definitions are needed. For a discrete (minimization) optimization problem, define the *solution space*, Ω, as a finite set of all possible solutions. Define an *objective function* f: $\Omega \rightarrow [0, +\infty]$ that assigns a non-negative value to each element of the solution space. Two important components of GHC algorithms are the *neighborhood function*, η: $\Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subseteq \Omega$ for all $\omega \in \Omega$, and the *hill climbing random variables* R_k: $\Omega \times \Omega \rightarrow \Re$, $k = 1,2,\ldots$. For each solution $\omega \in \Omega$, the neighborhood function $\eta(\omega)$ defines a set of solutions that are close to ω (Aarts and Korst 2002). The neighborhood function is assumed to be symmetric (i.e., if $\omega' \in \eta(\omega'')$, then $\omega'' \in \eta(\omega')$ for all $\omega',\omega'' \in \Omega$) and that $\omega \in \eta(\omega)$ for all $\omega \in \Omega$. Moreover, at each iteration of a GHC algorithm, a solution is randomly generated among all neighbors of the current solution by a neighborhood probability mass function, where the resulting random variables are independent (given the current solution). For example, neighbors are said to be generated uniformly at each iteration of a GHC algorithm execution if, for all $\omega \in \Omega$, with $\omega' \in \eta(\omega)$,

$$P\{\omega' \text{ is selected as the neighbor of } \omega \text{ at a given iteration of a GHC algorithm}\} = 1 / |\eta(\omega)|.$$

Without loss of generality, assume that if $\omega' \in \eta(\omega)$, then this probability is strictly positive.

The GHC algorithm is described below in pseudo-code. By definition, the hill climbing random variables, R_k, which are assumed to be independent, map points in $\Omega \times \Omega$ to distributions that determine whether a randomly generated neighboring solution is accepted during a particular inner loop iteration associated with outer loop iteration k. The stopping criterion for the inner loops, *STOP INNER*, determines when the hill climbing random variable index k increments by one, hence a new hill climbing random variable is used to accept or reject neighboring solutions. By setting the *STOP INNER* criterion to check whether the current solution is a local optimum, the hill climbing random variable changes only when a local optimum is visited; this will be further discussed below.

Although the range of the hill climbing random variables can be the set of real numbers, $\Re$, in practice they are typically restricted to the set of non-negative real numbers, $\Re^+$, (which is what will be assumed for the rest of the paper). Therefore, for minimization problems, when a randomly generated neighboring solution has objective function value greater than the current solution, then the neighboring solution is accepted (hence becomes the new current solution) if the difference between the objective function values is not too large (i.e., smaller than the value generated for the hill climbing random variable). This concept of accepting an inferior solution is the origin for the name "hill climbing".

Define a neighborhood function η and a set of hill climbing random variables R_k
Set the iteration indices $i = 0$, $k = 1$ and select an initial solution $\omega(0) \in \Omega$
Repeat
  Repeat
    Generate a neighboring solution $\omega \in \eta(\omega(i))$ and compute $\delta(\omega(i),\omega) = f(\omega) - f(\omega(i))$
    Generate an observation R from the random variable $R_k(\omega(i),\omega)$
    If $R \geq \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega$; Else (i.e., $R < \delta(\omega(i),\omega)$), then $\omega(i+1) \leftarrow \omega(i)$
    $i \leftarrow i+1$
  Until *STOP INNER*
  $k \leftarrow k+1$
Until *STOP OUTER*

Assume that the hill climbing random variables have finite means and finite variances (i.e., $E[|R_k(\omega(i),\omega)|] < +\infty$ and $Var[R_k(\omega(i),\omega)] < +\infty$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, $k = 1,2,\ldots$, $i = 1,2,\ldots$).

The neighborhood function establishes relationships between the solutions in the solution space, hence allows the solution space to be traversed or searched by moving between solutions. To ensure that the solution space is not fragmented, assume that all the solutions in the solution space (with neighborhood function η and neighborhood probability mass function $h(\omega)$) are *reachable* (i.e., for all $\omega',\omega'' \in \Omega$, there exists a set of solutions $\omega_1,\omega_2,\ldots,\omega_m \in \Omega$ such that $\omega_r \in \eta(\omega_{r-1})$, $r = 1,2,\ldots,m+1$, where $\omega' \equiv \omega_0$ and $\omega'' \equiv \omega_{m+1}$). If all solutions in the solution space are reachable, then the solution space (with neighborhood function η) is said to be reachable. Note that solution space fragmentation can be a problem, for example, in some implementations of tabu search with a deterministic tabu list. Fox (1993) describes a clever method on how

to avoid fragmentation altogether.

The objective function, f, and the neighborhood function, η, allow the solution space, Ω, to be decomposed into three mutually exclusive and collectively exhaustive sets:

- a set of global optima, $G = \{\omega^* \in \Omega: f(\omega^*) \leq f(\omega)$ for all $\omega \in \Omega\}$
- a set of local (but not global) optima, $L \equiv L(\eta) = \{\omega \in \Omega \backslash G: f(\omega) \leq f(\omega')$ for all $\omega' \in \eta(\omega)\}$
- a set of hill solutions, $H = \Omega \backslash (G \cup L)$.

Therefore $G \cup L$ are the set of local optima in Ω associated with neighborhood function η, where by definition, $\Omega = G \cup L \cup H$ with $G \cap L = \varnothing$, $G \cap H = \varnothing$, and $L \cap H = \varnothing$. Note also that for all $\omega \in G$, $\eta(\omega) \cap L = \varnothing$, and for all $\omega \in L$, $\eta(\omega) \cap G = \varnothing$ (i.e., a global optimum and a local optimum cannot be neighbors).

In practice, the best solution obtained over the entire GHC algorithm run, not just the final solution, is reported. This allows the algorithm to aggressively traverse the solution space visiting many inferior solutions en route to a globally optimal solution, while retaining the best solution obtained through the entire GHC run. By design, GHC algorithms are sampling procedures over the solution space Ω. For example, Monte Carlo search generates independent samples (with replacement) from the solution space, while simulated annealing (Henderson et al. 2003) generates samples guided by the neighborhood function, the objective function, and the temperature parameter. More specifically, simulated annealing can be described as a GHC algorithm by setting $R_k(\omega(i),\omega) = -t(k)ln(v_i)$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, $k = 1,2,\ldots$, where $t(k)$ is the temperature parameter (hence, defines a cooling schedule as $t(k) \rightarrow 0$) and $\{v_i\}$ are independent and identically distributed $U(0,1)$ random variables. Note that in the "accept improving or hill climbing moves" step of the GHC algorithm pseudo-code, for the simulated annealing hill climbing random variable, $R_k(\omega(i),\omega) \geq \delta(\omega(i),\omega)$ becomes $v_i \leq exp\{-\delta(\omega(i),\omega)/t(k)\}$, which is the standard form in which the simulated annealing hill climbing acceptance probability is described (Aarts and Korst 2002). Other algorithms that can be described using the GHC framework include threshold accepting (Dueck and Scheuer 1990), some simple forms of tabu search (Glover and Laguna 1997), Monte Carlo search, deterministic local search, the noising method (Charon and Hudry 2001), and Weibull accepting (see Jacobson et al. 1998 and Johnson and Jacobson 2002a,b for a discussion on how these algorithms can be fit into the GHC algorithm framework).

The iterations of a GHC algorithm can be classified using the concept of macro iterations. A *macro iteration* is a set of consecutive iterations that move the algorithm from any element of $G \cup L$ to any element of $G \cup L$ (including itself), where the solutions at any intervening iterations are (not necessarily distinct) elements of H. From the pseudo-code presented above, by requiring that the *STOP INNER* criterion checks whether the current solution is a local optimum, then the outer loops correspond to macro iterations. If there are a polynomial number of neighboring solutions of the current solution or the neighborhood of the current solutions can be searched in polynomial time, then verifying that the current solution is a local optimum can be done in polynomial time. Assume that this is the case, hence local optimality can be verified in polynomial time.

Using this *STOP INNER* criterion, at macro iteration k fixed, the iterations can be modeled as a homogeneous discrete-time Markov chain, with $|\Omega| \times |\Omega|$ *transition matrix*

$$P^k = \begin{bmatrix} P^k_{GG} & P^k_{GL} & P^k_{GH} \\ P^k_{LG} & P^k_{LL} & P^k_{LH} \\ P^k_{HG} & P^k_{HL} & P^k_{HH} \end{bmatrix},$$

where the entries of P^k denote the single iteration transition probabilities between all elements of Ω. Without loss of generality, assume that the GHC algorithm run is initialized at a solution $\omega(0) \in L$, since local search can be applied from any element in Ω, and the solution space is reachable. This places a restriction on the classes of discrete optimization problems that can be studied, since if a local optimum cannot be obtained in polynomial time in the size of the problem instance, then initializing the GHC algorithm run in this way may not be feasible (see Johnson et al. 1988, Jacobson and Solow 1993). In addition, if local search is applied and the local optimum obtained is a global optimum, then the problem is solved, though this may not be known until further iterations are executed.

In the pseudo-code presented above, for k fixed, the macro iterations can also be modeled as a homogeneous discrete-time Markov chain, with a $(|G|+|L|) \times (|G|+|L|)$ *macro iteration transition matrix,*

$$P^k_M = \begin{bmatrix} P^k_{GH}[\sum_{j=0}^{+\infty}(P^k_{HH})^j]P^k_{HG} + P^k_{GG} & P^k_{GH}[\sum_{j=0}^{+\infty}(P^k_{HH})^j]P^k_{HL} + P^k_{GL} \\ P^k_{LH}[\sum_{j=0}^{+\infty}(P^k_{HH})^j]P^k_{HG} + P^k_{LG} & P^k_{LH}[\sum_{j=0}^{+\infty}(P^k_{HH})^j]P^k_{HL} + P^k_{LL} \end{bmatrix},$$

where the entries represent the probability of a GHC algorithm moving from any element of $G \cup L$ to any element of $G \cup L$ (including itself), passing only through elements of H (Sullivan and Jacobson 2001). If P_{HH}^k is the zero matrix, then set $(P_{HH}^k)^0 \equiv \boldsymbol{I}$, the identity matrix. Matrix P_M^k can be simplified, since for all $\omega \in G$, $\eta(\omega) \cap L = \varnothing$, and for all $\omega \in L$, $\eta(\omega) \cap G = \varnothing$ (i.e., a global optimum and a local optimum cannot be neighbors), hence P_{GL}^k and P_{LG}^k are both zero matrices. Moreover, if a global optimum cannot be a neighbor of another global optimum, and a local optimum cannot be a neighbor of another local optimum, then P_{GG}^k and P_{LL}^k are both diagonal square matrices.

Consider a GHC algorithm applied to an instance of a discrete optimization problem. Assume that $R_k(\omega(i),\omega) \geq 0$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, for all outer loop, macro iterations $k = 1,2,\ldots$. At each macro iteration k, define the event

$$B(k) \equiv \{\text{The algorithm does not visit any element of } G \text{ over the first } k \text{ macro iterations}\} \quad (1)$$

and its complementary event

$$B^c(k) \equiv \{\text{The algorithm visits } G \text{ over the first } k \text{ macro iterations}\}. \quad (2)$$

By definition, $B(k) \supseteq B(k+1)$ for all macro iterations k, hence $\{B(k)\}$ is a telescoping, non-increasing sequence of events in k. Therefore, by the Monotone Convergence Theorem (Billingsley 1979),

$$P\{B(k)\} \to P\{B\} = P\{\bigcap_{k=1}^{+\infty} B(k)\} \text{ as } k \to +\infty. \quad (3)$$

Over the first k macro iterations, the algorithm visits k solutions, $\{\omega_1, \omega_2, \ldots, \omega_k\} \subseteq G \cup L$. Define f^k to be the minimum objective function value among these k solutions and ω^k to be the associated solution (i.e., $f^k = f(\omega^k)$ with $\omega^k = \text{argmin}\{f(\omega_j), j = 1,2,\ldots,k\}$). In practice, the best solution to date (i.e., ω^k) is reported. The key issue is whether $\omega^k \in G$. If $\omega^k \in G$, then the algorithm should be terminated no later than macro iteration k, while if $\omega^k \notin G$, then it would be desirable to determine whether the algorithm will at some future macro iteration visit a solution in G. Therefore, $P\{\omega^k \in G\} = P\{B^c(k)\}$ provides an algorithm performance measure for the solutions obtained within the first k macro iterations.

To establish the relationship between the convergence of a GHC algorithm and the event B, the following definition is needed.

Definition 1: A GHC algorithm *converges in probability* to G if $P\{C(k)\} \to 1$ as $k \to +\infty$, where $C(k) \equiv \{\omega_k \in G\} = \{\text{The algorithm is at an element of } G \text{ at macro iteration } k\}$.

Therefore, given an initial solution $\omega(0) \in L$, if a GHC algorithm converges in probability to G (as $k \to +\infty$), then $P\{B^c\}=1$. Equivalently, if $P\{B^c\} < 1$, then the algorithm does not converge in probability to G.

In light of these observations, the false negative problem asks whether a GHC algorithm will eventually visit G, given that the algorithm, after executing a finite number of macro iterations, has yet to visit G. The false negative probability is formally defined.

Definition 2: For a GHC algorithm, the *false negative probability* at macro iteration k is $P\{B^c \mid B(k)\}$, provided $P\{B(k)\} > 0$.

The false negative probability at macro iteration k provides a measure for the *effectiveness* of a GHC algorithm, namely the ability of an algorithm to visit G beyond macro iteration k. In particular, if $P\{B^c\}$ is small, then one can use the false negative probability to assess whether a GHC algorithm will eventually visit G; if the false negative probability at macro iteration k is sufficiently close to zero, then the algorithm may be terminated.

A necessary convergence condition for GHC algorithms can be obtained. To see this, recall that $P\{B(0)\} = 1$ (i.e., all GHC algorithm runs are initialized at an element of L). Furthermore, unless otherwise stated, assume that $P\{B^c(k)\} < 1$ for all macro iterations $k = 1,2,\ldots$.

For macro iteration k, define the conditional probability

$$r(k) \equiv P\{B^c(k) \mid B(k\text{-}1)\} = P\{C(k) \mid B(k\text{-}1)\}. \quad (4)$$

This probability can be used to quantify the false negative probability. Lemma 1 expresses the relationship between (4) and (1).

Lemma 1 (Jacobson and Yucesan 2004a)**:** Given a GHC algorithm initialized at solution $\omega(0) \in L$,

(i) $$P\{B(k)\} = \prod_{j=1}^{k} [1 - r(j)] \text{ for all macro iterations } k.$$

(ii) $$P\{B\} = \prod_{j=1}^{+\infty} [1 - r(j)].$$

Theorem 1 provides a closed form expression for the false negative probability.

Theorem 1 (Jacobson and Yucesan 2004a): Given a GHC algorithm initialized at solution $\omega(0) \in L$, for all macro iterations k with $P\{B(k)\} > 0$,

$$P\{B^c \mid B(k)\} = 1 - \prod_{j=k+1}^{+\infty} [1\text{-}r(j)]. \tag{5}$$

Theorem 2 provides upper and lower bounds for the false negative probability.

Theorem 2 (Jacobson and Yucesan 2004a): Given a GHC algorithm initialized at initial solution $\omega(0) \in L$, then for all macro iterations k with $P\{B(k)\} > 0$,

$$1 - exp\{-\sum_{j=k+1}^{+\infty} r(j)\} \le P\{B^c \mid B(k)\} \le 1 - exp\{-\sum_{j=k+1}^{+\infty} [r(j)] / [1 - r(j)]\}. \tag{6}$$

To compute the false negative probability for both convergent and nonconvergent GHC algorithms, Proposition 1 establishes the relationship between convergence in probability to G and visits to G in probability.

Proposition 1 (Jacobson and Yucesan 2004a): If a GHC algorithm converges in probability to G, then the GHC algorithm *visits G in probability* (i.e., $P\{B^c \mid B(k)\} = 1$ for all macro iterations $k = 1,2,\ldots$ with $P\{B(k)\} > 0$).

Proposition 2 provides necessary and sufficient conditions for a GHC algorithm to visit G in probability (i.e., the false negative probability is one for all macro iterations).

Proposition 2 (Jacobson and Yucesan 2004a): A GHC algorithm visits G in probability iff $\Sigma_{j=1,2,\ldots} r(j)=+\infty$.

Proposition 3 establishes the relationship between a GHC algorithm visiting G in probability and $P\{B^c\}$.

Proposition 3 (Jacobson and Yucesan 2004a): A GHC algorithm visits G in probability iff $P\{B^c\} = 1$.

Theorem 3 summarizes the relationship between $P\{B^c\}$, the false negative probabilities, $r(k)$, visits G in probability, and convergence in probability to G. It follows directly from Propositions 1, 2, and 3.

Theorem 3: Given a GHC algorithm initialized at initial solution $\omega(0) \in L$, consider the expressions

(D1) $P\{C(k)\} \to 1$ as $k \to +\infty$ (converges in probability to G).

(D2) $P\{B^c \mid B(k)\} = 1$ for all macro iterations k (visits G in probability).

(D3) $P\{B^c\} = 1$ (visits G in probability).

(D4) $\Sigma_{j=1,2,\ldots} r(j) = +\infty$ for all macro iterations k.

Then (D1) $\Rightarrow$ (D2) $\Leftrightarrow$ (D3) $\Leftrightarrow$ (D4).

Theorem 3 provides three necessary conditions for the convergence of a GHC algorithm. The only restriction on how the GHC algorithm traverses the solution space is that $P\{B(k)\} > 0$ for all macro iterations $k = 1,2,\ldots$. This restriction means that there is no finite-time convergence to G with probability one. Note that from Lemma 1, if $P\{B\} = \Pi_{j=1,2,\ldots} [1\text{-}r(j)] > 0$, then from Theorem 3, the GHC algorithm does not converge in probability to G. Moreover, since $C(k) \subseteq B^c(k)$ for all macro iterations $k = 1,2,\ldots$, then $P\{C(k)\} \le 1 - \Pi_{j=1,2,\ldots} [1\text{-}r(j)]$ for all macro iterations k.

Closed-form expressions for $r(k)$ can be obtained such that the results reported above can be applied to GHC algorithms, including the computation of the false negative probability and condition (D4) in Theorem 3. To obtain such an expression, the following definition is needed to represent $r(k)$ as a function of the macro iteration transition matrix, P_M^k.

Definition 3: For all $\omega \in L$, at macro iteration k, $Q(\omega,k) \equiv B(k) \cap$ {The algorithm is at solution ω at macro iteration k} with $q(\omega,k) = P\{Q(\omega,k)\}$.

Theorem 4 provides a closed-form expression for $r(k)$ in terms of the macro iteration transition matrix. This expression is used later to identify properties of non-convergent GHC algorithms.

Theorem 4 (Jacobson and Yucesan 2004a): Given a GHC algorithm initialized at initial solution $\omega(0) \in L$, then for all macro iterations k,

$$r(k) = \sum_{\omega_2 \in L} \sum_{\omega_1 \in G} q(\omega_2,k\text{-}1)\ P_{LH}^k(\omega_2,\bullet)\ [\sum_{j=0}^{\infty} \left(P_{HH}^k\right)^j]\ P_{HG}^k(\bullet, \omega_1). \tag{7}$$

The closed-form expression for $r(k)$ in Theorem 4 can be expressed in terms of the hill climbing random variable R_k. To see this, for all $\omega_1 \in G$, $\omega_2 \in L$, $\omega_3, \omega_4 \in H$,

$$P_{LH}^k(\omega_2,\omega) = \begin{cases} \left(1/|\eta(\omega_2)|\right) P\{R_k(\omega_2,\omega) \ge \delta(\omega_2,\omega)\}, & \omega \in \eta(\omega_2) \cap H \\ 0, & \omega \notin \eta(\omega_2) \cap H \end{cases} \tag{8}$$

and

$$P_{HH}^{k}(\omega_3,\omega_4)=\begin{cases}(1/|\eta(\omega_3)|)P\{R_k(\omega_3,\omega_4)\geq\delta(\omega_3,\omega_4)\}, & \begin{cases}\delta(\omega_3,\omega_4)>0\\ \omega_4\in\eta(\omega_3)\cap H\end{cases}\\ 1/|\eta(\omega_3)|, & \begin{cases}\delta(\omega_3,\omega_4)\leq 0\\ \omega_4\in\eta(\omega_3)\cap H\end{cases}\\ 0, & \begin{cases}\omega_3\in H\\ \omega_4\notin\eta(\omega_3)\cap H\end{cases}\end{cases} \quad (9)$$

Moreover, for all $\omega_1 \in G$, $\omega \in H$,

$$P_{HG}^{k}(\omega,\omega_1)=\begin{cases}(1/|\eta(\omega)|), & \omega_1\in\eta(\omega)\cap G\\ 0, & \omega_1\notin\eta(\omega)\cap G\end{cases} \quad (10)$$

To determine whether $\Sigma_{k=1,2,\ldots}\ r(k)$ converges, only the most dominant terms in (7) need to be considered (i.e., the terms in (7) that approach zero the slowest as $k \to +\infty$). From (8)-(10), the most dominant terms in (7) are $O(P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\})$ for $\omega_2 \in L$, $\omega \in \eta(\omega_2)\cap H$ (as $R_k \to_P 0$ as $k \to +\infty$). Therefore, if the hill climbing random variables are defined such that the probability of moving from any local optimum in a *single iteration* converges to zero sufficiently fast (as the number of macro iterations approaches infinity) so that the infinite sum (over k) of the $r(k)$ converges (i.e., (D4)), then from Theorem 3, the resulting GHC algorithm will not converge in probability to G. This necessary condition provides a simple feature to check for a given GHC algorithm, hence can be used to determine when a particular GHC does not converge.

To use this result in practice, $P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\}$, $\omega_2 \in L$, $\omega \in \eta(\omega_2)\cap H$, can be bounded above using the first and second moments of $R_k(\omega_2,\omega)$. For example, from Markov's inequality,

$$P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\}\leq E[R_k(\omega_2,\omega)]\,/\,\delta(\omega_2,\omega) \quad (11)$$

for $\omega_2 \in L$, $\omega \in \eta(\omega_2)\cap H$, (where $\delta(\omega_2,\omega)>0$). Moreover, by the one-sided Chebyshev inequality,

$$P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\}\leq Var[R_k(\omega_2,\omega)]\,/\,[Var[R_k(\omega_2,\omega)]+(\delta(\omega_2,\omega)-E[R_k(\omega_2,\omega)])^2]. \quad (12)$$

If either of these upper bounds approaches zero sufficiently fast such that $\Sigma_{k=1,2,\ldots}\ r(k) < +\infty$, then the GHC algorithm does not converge in probability to G.

To illustrate the use of these bounds, consider a simulated annealing algorithm with temperature parameters $t(k) = 1/k^2$. Then $E[R_k(\omega_2,\omega)] = 1/k^2$ and from (11), $P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\} \leq 1/(k^2\,\delta(\omega_2,\omega))$, which implies that $\sum_{k=1}^{+\infty} r(k) < +\infty$. Therefore, from Theorem 3, this simulated annealing algorithm does not converge in probability to G. On the other hand, for a simulated annealing algorithm with temperature parameters $t(k) = 1/k$, $E[R_k(\omega_2,\omega)] = 1/k$ and from (11), $P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\} \leq 1/(k\,\delta(\omega_2,\omega))$, which is not sufficient (from Theorem 3) to show that this simulated annealing algorithm does not converge in probability to G. However, $Var[R_k(\omega_2,\omega)] = 1/k^2$ and from (12), $P\{R_k(\omega_2,\omega)\geq\delta(\omega_2,\omega)\} \leq (1/k^2)/[1/k^2+(\delta(\omega_2,\omega)-1/k)^2]$ $= O((1/k\delta(\omega_2,\omega))^2)$ for k large, which implies that $\Sigma_{k=1,2,\ldots}\ r(k) < +\infty$. Therefore, from Theorem 3, this simulated annealing algorithm does not converge in probability to G.

The theoretical results described above can be used to assess the performance of various GHC algorithms. In particular, the performance of four GHC algorithms, Monte Carlo search, random-restart local search, threshold accepting, and simulated annealing can be evaluated.

Monte Carlo search is the process of randomly generating a large set of solutions in the solution space and taking the best solution among those generated. Theorem 3 implies that the false negative probability is one (for all macro iterations) for Monte Carlo search. To see this, Monte Carlo search can be described as a GHC algorithm by setting $\eta(\omega) = \Omega$ for all $\omega \in \Omega$, and $R_k = max\{|f(\omega)-f(\omega')|,\ \omega \in \Omega,\ \omega' \in \eta(\omega)\}$ for all macro iterations $k = 1,2,\ldots$. If $p(G) \equiv |G|\,/\,(|G|+|L|)$, then $r(k) = p(G)$. Therefore, $P\{B(k)\} = [1-p(G)]^k$. For macro iterations j and k, with $m > k$,

$$P\{B^c(m)\mid B(k)\} = 1-[1-p(G)]^{m-k},$$

which approaches one as $m \to +\infty$. Moreover, from Theorem 3, $\Sigma_{j=1,2,\ldots}\ r(j) = \Sigma_{j=1,2,\ldots}\ p(G) = +\infty$. This means that Monte Carlo search visits G in probability as $k \to +\infty$. However, $P\{C(k)\} = p(G)$ for all macro iterations k, hence Monte Carlo search does not converge in probability to G (i.e., from Theorem 3, (D2), (D3), and (D4) all hold, but (D1) is not satisfied).

Random-restart local search (or multi-start local search; see Marti 2003) combines Monte Carlo search and local search, by randomly selecting a new initial solution every time a local search algorithm terminates

at a local optimum. The analysis for Monte Carlo search also shows that the false negative probability is one (for all macro iterations) for random-restart local search, by redefining $p(G)$ to be the probability that a randomly generated initial solution in Ω will terminate at an element of G. Moreover, random-restart local search will not converge in probability to G (i.e., from Theorem 3, (D2), (D3), and (D4) all hold, but (D1) is not satisfied).

Threshold accepting is a particular GHC algorithm with $R_k(\omega(i),\omega) = t(k)$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, for macro iteration k, where $t(k) \to 0$ as $k \to +\infty$. Therefore, there exists $\varepsilon > 0$ sufficiently small and a macro iteration k_0 such that $|t(k)| < \varepsilon$ and $P\{R_k(\omega(i),\omega) \geq \delta(\omega(i),\omega)\} = 0$ for all $\omega(i) \in L$, $\omega \in \eta(\omega(i))$, and all $k \geq k_0$, hence (D4) in Theorem 3 does not hold. This implies that this common implementation of threshold accepting does not converge in probability to G. However, if $t(k)$ is set such that it does not approach zero, hence $r(k) \geq \delta$ for some $\delta > 0$ and for all macro iterations k, then (D4) in Theorem 3 may hold and the probability of a false negative is one at all macro iterations k. However, setting $t(k)$ in this way may not be feasible in practice, since it requires full knowledge of the solution space (with respect to the depth of all local and global optima; see Hajek 1988). Although the given formulation of threshold accepting does not converge in probability to G, it often yields satisfactory results in practice (Franz et al. 2001, Abboud et al. 1998, Nissen and Paul 1995). This observation further supports the belief that asymptotic convergence is not necessarily a good predictor of finite-time performance.

Simulated annealing is a particular GHC algorithm with $R_k(\omega(i),\omega) = -t(k)ln(\nu_i)$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, $k = 1,2,\ldots$, where $t(k)$ is the temperature parameter (hence, defines a cooling schedule as $t(k) \to 0$) and $\{\nu_i\}$ are independent and identically distributed $U(0,1)$ random variables. The necessary condition (D4) in Theorem 3 can be related to the convergence conditions for simulated annealing presented in Hajek (1988). In particular, Hajek (1988) shows that simulated annealing converges in probability to a global optimum if and only if $\Sigma_{k=1,2,\ldots} e^{-(d^*/t(k))} = +\infty$, where the temperature parameters $t(k)$ define a nonincreasing cooling schedule (that approaches zero as $k \to +\infty$), and d^* is the maximum depth of all local optima (i.e., the maximum gap in objective function value between an element of L and the solution in H that can reach an element of G via local search, where the maximum is taken over all elements of L). This result assumes that the depth of all elements in G is infinity, hence once a global optimum is reached, simulated annealing cannot escape from it (with probability one). Since the neighborhood function η is defined such that the solution space is reachable, then at each macro iteration k that is sufficiently large, there is a positive probability that the algorithm will need to escape from each element of L and move to an element of G. In particular, at each macro iteration k sufficiently large, the conditional probability $r(k)$ has a component that includes the probability of escaping from the deepest local optimum. Therefore, using the law of total probability,

$$r(k) = \Sigma_{\omega \in L}\, r(k \mid \omega \in L \text{ is visited at macro iteration } k\text{-}1\}\, P\{\omega \in L \text{ is visited at macro iteration } k\text{-}1\}$$

Therefore, there exists a lower bound for $r(k)$ that is a linear function of $P\{$moving from the deepest element of L to an element of $G\}$ = P{Accepting hill climbing moves out of the deepest element of L to an element of $G\} = O(e^{-(d^*/t(k))})$, since the hill climbing random variable at macro iteration k is exponential with mean $1/t(k)$. Therefore, if $\Sigma_{k=1,2,\ldots} e^{-(d^*/t(k))} = +\infty$, then condition (D4) in Theorem 3 is satisfied.

Another consequence of Theorem 3 is that different simulated annealing algorithms may not converge in probability to G, but they may visit G in probability. For example, fixed temperature implementations of simulated annealing are provably non-convergent (since the temperature parameter does not approach zero), but visit G in probability (since $r(k) > \varepsilon > 0$ for all k for some ε fixed). Cohn and Fielding (1999) and Fielding (2000) present interesting theoretical and empirical results suggesting that there is an optimal fixed temperature for simulated annealing for different classes of problems. Orosz and Jacobson (2002a,b) also present results with fixed temperature simulated annealing algorithms, including analytical expressions for the expected number of iterations needed to reach a prespecified objective function value. The results in Theorem 3 are consistent with the observations in Cohn and Fielding (1999) and Fielding (2000). Moreover, from Lemma 1, the rate at which $\Pi_{j=1,2,\ldots,k}\,[1 - r(j)]$ converges to zero as $k \to +\infty$ (or equivalently, the rate at which $\Sigma_{j=1,2,\ldots,k}\, r(j)$ diverges to infinity as $k \to +\infty$) provides a measure for comparing two fixed temperature simulated annealing algorithms.

2. Tabu Guided Generalized Hill Climbing Algorithms

An accomplishment during the term of this grant has been the development of a mathematical framework for combining tabu search and generalized hill climbing algorithms. Vaughan and Jacobson (2004) formulate tabu search strategies that guide generalized hill climbing (GHC) algorithms for addressing NP-hard discrete optimization problems. The resulting framework, termed tabu guided generalized hill climbing (TG^2HC) algorithms, uses a tabu release parameter that probabilistically accepts solutions currently on the tabu list. TG^2HC algorithms are modeled as a set of stationary Markov chains, where the tabu list is fixed for each outer loop iteration. This framework provides practitioners with guidelines for developing tabu search strategies to use in conjunction with GHC algorithms that preserve some of the algorithms' known

performance properties. In particular, sufficient conditions are obtained that indicate how to design iterations of problem-specific tabu search strategies, where the stationary distributions associated with each of these iterations converge to the distribution with zero weight on all non-optimal solutions.

Tabu search strategies (Glover and Laguna 1997) provide powerful tools for addressing hard discrete optimization problems. These strategies can be used to effectively guide local search algorithms in search of optimal/near-optimal solutions by developing and updating a tabu list of forbidden moves of neighboring solutions. Therefore, tabu search strategies are often described as *meta-heuristics* that guide local search algorithms to generate solutions that the local search algorithm, applied independently, would not have otherwise visited.

The wide variety of both tabu search strategies and local search algorithms make it difficult to analytically evaluate the performance of such strategies. Consequently, there are a limited number of results in the literature that rigorously quantify their performance. Aarts and Lenstra (1997) state

> "Tabu search is a general scheme that must be tailored to the details of the problem at hand. Unfortunately, there is little theoretical knowledge that guides the tailoring process, and users have to resort to the available practical experience."

Therefore the utility and performance (e.g., convergence properties) of tabu search strategies is typically studied on a case-by-case basis. Moreover, the decision of when to apply tabu search strategies often relies on experimentation and explicit knowledge of a particular problem. Consequently, tabu search strategies are often poorly applied, resulting in blocked access to large sections of a problem's solution space that may contain optimal/near optimal solutions.

The generalized hill climbing (GHC) algorithm framework (Jacobson et al. 1998) provides a structure for modeling local search algorithms to address intractable discrete optimization problems. GHC algorithms are designed to find optimal solutions for discrete optimization problems by permitting visits to inferior solutions enroute to optimal/near optimal solutions. The GHC algorithm framework includes several well-known local search algorithms as special cases.

Vaughan and Jacobson (2004) formulates tabu search strategies that guide GHC algorithms. The resulting framework, termed tabu guided generalized hill climbing (TG^2HC) algorithms, is neither problem specific nor GHC algorithm specific. TG^2HC algorithms incorporate a tabu release parameter that allows the algorithm to probabilistically accept solutions currently on the tabu list according to an iteration dependent function that controls the probability that solutions on the tabu list will be visited. By design, numerous tabu search strategies can be modeled within the TG^2HC algorithm framework.

Vaughan and Jacobson (2004) shows that TG^2HC algorithms can be modeled as a set of stationary Markov chains, where the tabu list is fixed for each outer loop iteration. During each outer loop iteration, a set of inner loop iterations is performed, where information about the solution space is used to define the tabu list for the subsequent outer loop iteration. Therefore, each outer loop iteration (with its associated set of inner loop iterations) can be modeled as a Markov chain, though the set of outer loop iterations cannot be modeled as a Markov Chain. Moreover, the TG^2HC algorithm framework provides practitioners with guidelines for developing tabu search strategies to use in conjunction with GHC algorithms that preserve such algorithms' known performance properties.

Modeling TG^2HC algorithms in this way allows the performance of tabu search strategies to be studied within a general framework. For a TG^2HC algorithm, the resulting Markov chain transition matrices can be written in terms of the transition matrices corresponding to the inner loop iterations of a GHC algorithm without the tabu search strategy. Therefore, by modeling a TG^2HC algorithm as a set of Markov chains, properties that are known for GHC algorithm can be extended to TG^2HC algorithms. Moreover, TG^2HC algorithms provide practitioners with guidelines for developing tabu search strategies to use in conjunction with a particular GHC algorithm that do not destroy such algorithm's known performance properties.

As described previously, applying a local search algorithm typically requires the formulation of a neighborhood function. At each iterations of a local search algorithm, a neighboring solution of the current solution is generated by a *neighborhood probability mass function* for the neighborhood function η, $g_k(\omega_i, \omega') = P(\omega' \in \eta(\omega_i)$ is generated during iteration $k)$.

Theorem 5 shows that a GHC algorithm applied to a discrete optimization problem can be modeled as a stationary (discrete-time) Markov chain (Isaacson and Madsen 1985).

Theorem 5: A GHC algorithm applied to a discrete optimization problem with solution space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ can be modeled by a stochastic process $\{\overline{Q}_n^k\}$, $k = 1,2,\ldots,K$, $n = 1,2,\ldots,N(k)$, $\overline{Q}_n^k \in \Omega$ with state space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ that satisfies the Markov property for all inner loop iterations n and all states $\omega_1, \omega_2, \ldots, \omega_n$ (i.e., $\{\overline{Q}_n^k\}$ is a Markov chain). Moreover, for all $k = 1,2,\ldots,K$, the stationary Markov chain has corresponding transition matrices $\overline{P}(k)$, where

$$\overline{P}_{ij}(k)=\begin{cases} g_k(\omega_i,\omega_j)P(R_k(\omega_i,\omega_j)\geq\delta_{ij}) & \text{for all } \omega_i\in\Omega,\ \omega_j\in\eta(\omega_i),\ j\neq i \\ 1-\sum\limits_{\substack{z\in\eta(\omega_i)\\ z\neq i}}\overline{P}_{iz}(k) & j=i \\ 0 & \text{otherwise} \end{cases} . \qquad (13)$$

Proof: Obtained directly from results and observations in Johnson and Jacobson (2002a).

Tabu guided generalized hill climbing algorithms are now described. Vaughan and Jacobson (2004) shows that each outer loop iteration (which represents a set of inner loop iterations) of a tabu guided generalized hill climbing algorithm can be modeled by a stationary Markov chain, where the corresponding transition matrices can be written in terms of the transition matrix $\overline{P}(k)$ in (13).

The term tabu search was first introduced and coined by Glover (1977, 1986). Tabu search strategies (Glover and Laguna 1997) can be used to effectively guide local search algorithms towards optimal/near-optimal solutions, by developing and updating a *tabu list* (i.e., a set of forbidden moves). Glover and Laguna (1997) describe numerous tabu search strategies.

A *meta-heuristic* (Glover 1986) is a strategy that guides one or several heuristics to visit solutions that the heuristics would not have otherwise explored. There are no restrictions limiting the type of heuristic that a meta-heuristic guides. The heuristic is often a local search algorithm (such as simulated annealing). For example, Faigle and Kern (1992) discuss probabilistic tabu search as a meta-heuristic that guides simulated annealing. The convergence results in Faigle and Kern (1992) are based on modeling simulated annealing in terms of Markov processes (Faigle and Schrader 1988). Glover (1989) provides a comparison between probabilistic tabu search and simulated annealing.

Tabu search strategies can be viewed as manipulations of the neighborhood function for local search algorithms at a given iteration (based on information obtained during previous iterations), with the objective of improving performance. Tabu search strategy decisions may be based on either explicit and/or attributive memory (Glover and Laguna 1997). *Explicit memory* records entire solutions, whereas *attributive memory* keeps track of solution attributes (i.e., a characteristic associated with a particular solution). Since an entire solution can be described as an attribute, it is sufficient to refer to the elements on a tabu list as solution attributes. The neighborhood restrictions (or enhancements) imposed by tabu search strategies may facilitate one or more objectives, such as avoiding cycling through the same set of solutions, escaping local optima, and exploring particular areas of the solution space (Glover 1977). Note that even when the objective of implementing a tabu search strategy is clear, what to place in a tabu search strategy memory structure to accomplish this objective may be difficult to determine. Therefore, *candidate list strategies* are used to define the tabu list and how it is updated.

Glover and Laguna (1997) discuss the four tabu search dimensions: influence, quality, recency, and frequency. These may be thought of as incentive dimensions for tabu search strategy memory structures. A memory structure from the influence dimension evaluates the effect of choices made during the search. When the quality of a solution attribute is considered in decision-making, a quality memory structure is being applied. Quality can be used to catalog solution attributes that are shared by good/reasonable solutions. Recency memory structures are based on the solution attributes of recently visited solutions. Frequency memory structures are used to make decisions based on frequencies associated with solution attributes. Note that these four memory structures are not mutually exclusive and can be employed simultaneously. Moreover, each solution attribute on a tabu list is assigned a *tabu tenure* (i.e., the number of iterations that a solution attribute remains on the tabu list). It is not necessary that every solution attribute on the tabu list have the same tabu tenure. Moreover, tabu tenure can be either finite or infinite.

Vaughan and Jacobson (2004) describes a framework for modeling tabu search strategies that guide GHC algorithms for addressing discrete optimization problems. The resulting framework, termed *tabu guided generalized hill climbing* (TG^{2}HC) algorithms, places no restrictions on the design of the tabu list, hence it can include numerous tabu search strategies guiding GHC algorithms.

Tabu search strategies are designed to partition the solution space. To describe how this is done, consider a fixed iteration of a TG^{2}HC algorithm. The TG^{2}HC algorithm framework builds a tabu list T based on any of the four tabu search dimensions or a combination thereof. TG^{2}HC algorithms are designed such that T does not change (i.e., is fixed) during all the inner loop iterations associated with an outer loop iteration. Therefore, T is only updated at the end of the outer loop iterations based on the set of solutions visited during the inner loop iterations. Note that tabu strategies that update T at every iteration can be modeled in the TG^{2}HC algorithm framework by executing only a single inner loop iteration. However, the Markovian property of the inner loop iterations (for each fixed outer loop iteration) would no longer be satisfied, since this property is only satisfied when the number of inner loop iterations is large. Nonetheless, the structure of the TG^{2}HC algorithm framework can still be used, since it provides a general framework for how to imbed

tabu search strategies within GHC algorithms.

The key difference between TG^{2}HC and GHC algorithms is the incorporation of T and a tabu release parameter, $\Phi(k, \omega(i), \omega)$, where $0 \le \Phi(k, \omega(i), \omega) \le 1$ (see Section 4.4). In fact, comparing GHC algorithm pseudo-code with TG^{2}HC algorithm pseudo-code given below, the only difference is that in each inner loop iteration of the TG^{2}HC algorithm, if the neighboring solution generated is on T and rejected based on the tabu release parameter value, then this solution is not given the opportunity to be accepted as the new current solution.

TG^{2}HC Algorithm

Set the outer loop counter bound K and the inner loop counter bounds $N(k)$, $k = 1,2,\ldots,K$
Define a set of hill climbing random variables $-\infty \le R_k \le +\infty$, $k = 1,2,\ldots,K$
Set the iteration indices $i = 0$, $k = n = 1$
Generate an initial solution $\omega(0) \in \Omega$ and initialize the tabu list T
Repeat while $k \le K$
  Repeat while $n \le N(k)$
    Generate a solution $\omega \in \eta(\omega(i))$ using neighborhood probability mass function $g_k(\omega(i),\omega)$
    If $\omega \in T$, then generate $u = U(0,1)$. If $u > \Phi(k, \omega(i), \omega)$, go to *
    Generate an observation R from $R_k(\omega(i), \omega)$
    Compute $\delta(\omega(i), \omega) = f(\omega) - f(\omega(i))$
    If $R \ge \delta(\omega(i), \omega)$, set $\omega(i+1) \leftarrow \omega$; Else (i.e., $R < \delta(\omega(i), \omega)$), set $\omega(i+1) \leftarrow \omega(i)$
  * $n \leftarrow n+1$, $i \leftarrow i+1$
  Until $n = N(k)$
  Update the tabu list, T
  $n = 1$, $k \leftarrow k+1$
Until $k = K$

TG^{2}HC algorithms incorporate a tabu release parameter, $\Phi(k, \omega(i), \omega)$, that allows solutions on T to be accepted according to an iteration dependent function. Therefore, the tabu release parameter controls the probability that solutions on the tabu list are visited. Candidate solutions with solution attributes on the tabu list are released from their tabu status at outer loop iteration k with probability $0 \le \Phi(k, \omega(i), \omega) \le 1$. Since TG^{2}HC algorithms place no restrictions on how T is updated, any tabu search strategy can be modeled within the TG^{2}HC algorithm framework by setting the parameter $\Phi(k, \omega(i), \omega) = 0$ for every $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, and for all $k = 1,2,\ldots$.

The tabu release parameter can also be used to define a new type of probabilistic tabu search strategy that is distinct from probabilistic tabu search (Glover 1989, Glover and Laguna 1997). This new probabilistic tabu search strategy can be defined by setting $\Phi(k, \omega(i), \omega') = g_k(\omega(i), \omega')$. Therefore, if a tabu neighboring solution $\omega' \in \eta(\omega(i)) \cap T$ is generated, then ω' is accepted with probability $\Phi(k, \omega(i), \omega') P(R_k(\omega(i), \omega') \ge \delta(\omega(i), \omega'))$ (see Section 4.5). Glover (1989) discusses a nonsequential approach to determining the transition probabilities, where each element $\omega' \in \eta(\omega(i))$ is assigned a weight $w(\omega')$ that is determined using tabu search strategies. For this case,

$$g_k(\omega(i),\omega') = P(\text{Generating } \omega' \in \eta(\omega(i))) = w(\omega') \Big/ \sum_{\omega \in \eta(\omega(i))} w(\omega) \quad .$$

Define *probabilistically decreasing tabu search* as a meta-heuristic for GHC algorithms, where for all $\omega(i)$, $\omega \in \Omega$ and for all $k = 1,2,\ldots$, $0 < \Phi(k, \omega(i), \omega) < 1$, where $\Phi(k, \omega(i), \omega)$ is monotonically increasing and $\lim_{k \to \infty} \Phi(k, \omega(i), \omega) = 1$. Theorem 6 below shows that a Markov chain model can be formulated only for the inner loop iterations of T^{2}GHC algorithms (not for the outer loop iterations).

Theorem 6 (Vaughan and Jacobson 2004): For each outer loop iteration, the inner loop iterations of a TG^{2}HC algorithm applied to a discrete optimization problem with solution space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ can be modeled by a stochastic process $\{Q_n^k\}$, $k = 1, 2, \ldots, K$, $n = 1, 2, \ldots, N(k)$, $Q_n^k \in \Omega$ with state space $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ that satisfies the Markov property for every n and all states $\omega_1, \omega_2, \ldots, \omega_n$ (i.e., $\{Q_n^k\}$ is a Markov chain). Moreover, the stationary Markov chain has corresponding transition matrices

$$P_{ij}(k)=\begin{cases}\overline{P}_{ij}(k) & \text{for all } \omega_i\in\Omega,\ \omega_j\in\eta(\omega_i)\cap T^c, j\neq i\\ \Phi(k,\omega_i,\omega_j)\overline{P}_{ij}(k) & \text{for all } \omega_i\in\Omega,\ \omega_j\in\eta(\omega_i)\cap T, j\neq i\\ 1-\sum_{\substack{z\in\eta(\omega_i)\\ z\neq i}} P_{iz}(k) & j=i\\ 0 & \text{otherwise}\end{cases} \tag{14}$$

where $\overline{P}_{ij}(k)$ is the transition matrix that corresponds to the Markov chain that models the underlying GHC algorithm for the TG^{2}HC algorithm.

The TG^{2}HC algorithm Markov chain model allows the performance of tabu search strategies to be studied within a general framework. Moreover, insights into how to implement tabu search strategies can be obtained from this framework. For example, modeling TG^{2}HC algorithms as Markov chains allows properties that are known for particular GHC algorithms to be extended to TG^{2}HC algorithms. Moreover, the TG^{2}HC algorithm framework provides guidelines for developing tabu search strategies to use in conjunction with a GHC algorithm that do not affect the algorithm's known performance properties.

Theorem 8 provides conditions under which a TG^{2}HC algorithm will not affect the performance properties of the underlying GHC algorithm. In particular, given a GHC algorithm with neighborhood probability mass function g_k and transition matrices $\overline{P}(k)$, Theorem 8 provides sufficient conditions for a unique stationary distribution to exist at each outer loop iteration k, and conditions that guarantee that the limit of these stationary distributions contains zeros for all non-optimal solutions (Johnson and Jacobson 2002a). The following result (from Johnson and Jacobson 2002a) is needed to prove Theorem 8.

Theorem 7: (Johnson and Jacobson 2002a): Consider a GHC algorithm applied to an instance of a discrete optimization problem (Ω, f) with neighborhood function η. Define the GHC neighborhood probability mass function by g_k and the hill climbing random variables by $\{R_k\}$, resulting in transition probabilities $\overline{P}_{ij}(k)$. Assume that g_k and $\overline{P}_{ij}(k)$ satisfy:

(a) For all $\omega_i, \omega_j \in \Omega$ and all outer loop iterations k, there exists a positive integer d and a corresponding sequence of solutions $l_0, l_1, \ldots, l_d \in \Omega$ with $l_0 = \omega_i$, $l_d = \omega_j$, and $g_{l_m, l_{m+1}}(k) > 0$, $m = 0,1,..,d\text{-}1$,

(b) for all $\omega_i, \omega_j \in \Omega$, $\omega_j \in \eta(\omega_i)$, and all outer loop iterations k, $\lim_{k\to\infty} g_k$ exists and is strictly positive.

Moreover, assume that the acceptance probabilities satisfy

(c) $P(R_k \geq \delta) > 0$ for all $\omega_i \in \Omega$, $\omega_j \in \eta(\omega_i)$, and all outer loop iterations k,

(d) $f(\omega_i) < f(\omega_j) \Rightarrow \lim_{k\to\infty} P(R_k \geq \delta) = 0$.

Then the stationary distribution $\pi(k)$ exists for each outer loop iteration k. Moreover, $\lim_{k\to\infty} \pi_i(k) = 0$ for all $\omega_i \in \Omega$ that are neither local or global optima, provided that the limit exists.

Conditions (a) and (b) are, by design, straightforward to satisfy. For example, if the neighborhood probability mass function is not a function of the outer loop iteration k, and if for all $\omega_i, \omega_j \in \Omega$, $g_k(\omega(i), \omega(j))$ is strictly positive, then conditions (a) and (b) hold. Condition (c) requires that for all $\omega_i \in \Omega$, the probability of accepting every neighboring solution $\omega_j \in \eta(\omega_i)$, for all iterations k, is strictly positive. Lastly, condition (d) implies that as the outer loop iterations approach infinity, the probability of accepting neighbors of higher objective function value approaches zero.

Note that the proof of Theorem 7 in Johnson and Jacobson (2002a) depends only on the inner loop iterations (associated with an outer loop iteration) satisfying the Markov property. Theorem 8 shows that if a GHC algorithm satisfies certain conditions, then a probabilistically decreasing tabu search strategy can be defined that guides the GHC algorithm, and performance results can be obtained for the resulting TG^{2}HC algorithm.

Theorem 8 (Vaughan and Jacobson 2004): Consider a GHC algorithm applied to an instance of a discrete optimization problem (Ω, f) with neighborhood function η. Define the GHC neighborhood probability mass function by g_k and the hill climbing random variables by $\{R_k\}$, resulting in transition probabilities $\overline{P}_{ij}(k)$. Assume that $\overline{P}_{ij}(k)$ satisfy (a)-(d) in Theorem 3. Consider a probabilistically decreasing TG^{2}HC algorithm applied to an instance of (Ω, f), where the same neighborhood probability mass function g_k and hill climbing random variables $\{R_k\}$ are considered. Define the resulting transition matrix by $P(k)$. Then $\pi(k)$ exists for each k. Moreover, if $\lim_{k\to\infty} \pi(k)$ exists, then $\lim_{k\to\infty} \pi_i(k) = 0$ for all $\omega_i \in \Omega$ that are neither local or global optima.

Since the TG^{2}HC algorithm framework allows different tabu search strategies to be incorporated into GHC

algorithms (e.g., simulated annealing), Theorem 8 provides guidelines for developing tabu search strategies to use in conjunction with such algorithms that preserve their performance properties. For example, probabilistic tabu search (Faigle and Kern 1992) incorporates memory into simulated annealing to enhance its performance. TG^{2}HC is designed in a similar manner by incorporating memory into GHC algorithms. Moreover, Theorem 8 provides sufficient conditions that define how iterations of problem-specific tabu search strategies should be designed such that the stationary distributions of these iterations converge to the distribution with total weight zero on the non-optimal solutions. For example, any tabu release parameter that satisfies the conditions that $0 < \Phi(k, \omega_i, \omega_j) < 1$ is monotonically increasing and $\lim_{k\to\infty} \Phi(k,\omega_i,\omega_j) = 1$ are sufficient for Theorem 8 to hold, hence the performance of the GHC algorithm is preserved. Therefore, $\Phi(k, \omega_i, \omega_j) = 1 - (1/k)$ for all $\omega_i \in \Omega$, $\omega_j \in \eta(\omega_i) \cap T$ satisfies these conditions, while $\Phi(k, \omega_i, \omega_j) = \frac{1}{2}$ for all $\omega_i \in \Omega$, $\omega_j \in \eta(\omega_i) \cap T$ does not.

The TG^{2}HC framework includes both tabu search and GHC algorithms as special cases. In particular, when $\Phi(k, \omega_i, \omega_j) = 0$ for all $\omega_i \in \Omega$, $\omega_j \in \eta(\omega_i) \cap T$, then the resulting TG^{2}HC algorithm reduces to tabu search with tabu list T, while when $\Phi(k, \omega_i, \omega_j) = 1$ for all $\omega_i \in \Omega$, $\omega_j \in \eta(\omega_i) \cap T$, the resulting TG^{2}HC algorithm reduces to GHC. This suggests that the tabu release parameters acts as a weight that determines the fraction of tabu search and the fraction of GHC that is being using during a TG^{2}HC algorithm execution, analogous to taking a convex combination of two algorithms, where $\lim_{k\to\infty} \Phi(k,\omega_i,\omega_j) = 1$ translates into a growing amount of weight being shifted from tabu search to the GHC algorithm as the TG^{2}HC algorithm executes. Therefore, the TG^{2}HC algorithm framework provides a natural generalization of probabilistic tabu search by moving beyond the acceptance criteria incorporated in simulated annealing and by allowing for the tabu list to be overridden probabilistically, with the requirement that the tabu status of solutions on the tabu list be weakened as the algorithm executes.

The sufficient conditions in Theorem 8 are an extension of results in the literature for local search algorithms (Johnson and Jacobson 2002a) to TG^{2}HC algorithms, by showing how a Markov chain model can be used to describe the long run behavior of such algorithms. Moreover, these conditions show how properties that are known for local search algorithms can be extended to TG^{2}HC algorithms. By modeling TG^{2}HC algorithms as Markov chains, the performance of tabu search strategies can be studied within a general framework. These results also show how TG^{2}HC algorithms generalize results for GHC algorithms (Johnson and Jacobson 2002a).

3. Simultaneous Generalized Hill Climbing Algorithms

An accomplishment during the term of this grant has been the completion of the study and investigation of simultaneous generalized hill climbing algorithms. These results have been reported in Vaughan et al. (2005, 2007), and applied to a combat search-and-rescue operation problem reported in Jacobson et al. (2006b).

Simultaneous generalized hill-climbing (SGHC) algorithms were introduced as a general framework for simultaneously addressing a set of related discrete optimization problems using heuristics. It is common to encounter several discrete optimization problems where a relationship exists between the solution spaces of the individual problems. In general, these problems are addressed individually. Because of their similarities, the same computational tools can be effectively used to address them. Therefore, the traditional approach has been to address each discrete optimization problem individually using the same heuristic. For example, in the late 1990's, the Material Process Design Branch of the U.S. Air Force Research Laboratory, Wright Patterson Air Force Base (Dayton, Ohio) had several similar discrete optimization problems under study. Each discrete optimization problem was a manufacturing-process design optimization problem (design sequence). The large number of design sequences and associated input-parameter-setting combinations made this set of problems difficult to solve. Several heuristics in the generalized hill-climbing algorithm (GHC) framework were introduced to address such manufacturing design problems (Jacobson et al. 1998) individually. The SGHC algorithm framework was motivated by the results reported in Vaughan et al. (2000), which developed a new neighborhood function that allows a heuristic to identify a near-optimal discrete manufacturing design sequence among a set of valid design sequences. This neighborhood function allows heuristics simultaneously to optimize over both the design sequences *and* the input parameters, eliminating the need to address each design sequence (i.e., each discrete optimization problem) individually. Therefore, all the design sequences were addressed in a single algorithm execution. Moreover, during the algorithm's execution, the new neighborhood function allowed information gained while optimizing over the current design sequence to be used to define the initial input parameters (solution) for the subsequent design sequence. This was accomplished by considering common processes across these two design sequences and retaining the optimal parameter values over the current design sequence to generate the initial solution in the subsequent design sequence. The computational results in Vaughan et al. (2000) suggest that such an approach is not only feasible but yields results that are superior to the previous approach of addressing each design sequence individually. The limitation of this approach is that the neighborhood function is problem specific.

This research completes the research analysis initially reported in Vaughan et al. (2000) by formally defining a class of sets of related discrete optimization problems, similar to the one described for the manufacturing design problem. A set of discrete optimization problems contained in this class is defined as a set of *fundamentally related* discrete optimization problems. Informally, a set of discrete optimization problems are fundamentally related if they share a common objective function and have some well-defined overlap.

The SGHC algorithm framework is used to simultaneously to address sets of fundamentally related discrete optimization problems using heuristics. A metric (i.e., distance measure) between elements in a set of fundamentally related discrete optimization problems is introduced. This metric is a measure of the overlap between two discrete optimization problems. SGHC algorithms probabilistically move between elements in a set of fundamentally related discrete optimization problems during their execution according to a problem probability mass function that depends on both the iteration counter and the metric. Therefore, an SGHC algorithm can be defined such that movement between discrete optimization problems that are significantly similar occurs more frequently than movement between discrete optimization problems that are less similar (hence benefit less from an exchange of information). When an SGHC algorithm moves between discrete optimization problems, information gained while optimizing over the current discrete optimization problem is used to set the initial solution in the subsequent discrete optimization problem. The information used is determined by the practitioner, for the particular set of problems under study. However, effective strategies are often apparent based on the problem description.

The SGHC framework is developed such that it can be applied to a wide variety of sets of related manufacturing, military, and service-industry discrete optimization problems. For example, the algorithm introduced in Vaughan et al. (2000) can be described as a SGHC algorithm for addressing a set of discrete-manufacturing process-design optimization problems. Three additional examples of sets of fundamentally related discrete optimization problems are described in Vaughan et al. (2005): a set of traveling-salesman problems (TSPs), a set of permutation flow-shop problems, and a set of Max-Satisfiability problems. These illustrative examples suggest how other sets of discrete optimization problems can be modeled such that the SGHC algorithm can be easily implemented.

Vaughan et al. (2005) present new computational results using the SGHC algorithm to address randomly generated problems of the illustrative examples. For each set of discrete optimization problems, a heuristic is embedded in the SGHC algorithm framework (either simulated annealing or local search). For comparison purposes, the same heuristics are also applied individually to each discrete optimization problem in the sets. The computational results suggest that SGHC algorithms outperform the heuristics applied individually to the discrete optimization problems in the set, as measured by the average optimal solution across 30 independent replications. Jacobson et al. (2006b) report computational results using SGHC algorithms to address a specific instance of a set of TSPs. Moreover, Vaughan et al. (2000) illustrate the advantages of approaching a set of discrete-manufacturing process-design optimization problems using SGHC algorithms.

To describe the SGHC algorithm framework, several definitions are needed. Define a discrete optimization problem by a finite set of solutions, $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$, together with an *objective function* f: $\Omega \rightarrow \Re$. Without loss of generality, assume that all discrete optimization problems are minimization problems. To address NP-hard discrete optimization problems, heuristics are formulated with the goal of identifying near-optimal solutions. To apply a heuristic to such problems, define a *neighborhood function*, η: $\Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subset \Omega$ for all $\omega \in \Omega$. Note that the neighborhood function maps each solution $\omega \in \Omega$ to a set of solutions $\eta(\omega) \subset \Omega$ and that this set is contained in the power set of Ω (i.e., $\eta(\omega) \in 2^{\Omega}$). Define $g_{pj}(i)$ to be the *solution probability mass function* for the neighborhood function η, namely, the probability that $\omega_j \in \eta(\omega_p)$ is generated during iteration i.

It is often necessary, in practice, to approach sets of similar discrete optimization problems. Limited progress has been made in designing a framework for exploiting the similarities between such sets of problems. For example, Zhang and Dietterich (1997) introduce a methodology for solving discrete optimization problems through the application of reinforcement learning, where information obtained while solving one discrete optimization problem can be used to determine reasonable parameters for solving other similar problems.

The GHC algorithm framework (Jacobson et al. 1998) provides a structure for using heuristics to address intractable discrete optimization problems. The GHC algorithm framework includes numerous heuristics that seek to find optimal solutions for discrete optimization problems by allowing the algorithm to visit inferior solutions en route to an optimal or near optimal solution. All GHC algorithms are formulated using three components, a set of *hill-climbing random variables* $\{R_k\}$, a neighborhood function η, and a solution probability mass function g. This structure permits exploration into the behavior of families of GHC algorithms. GHC algorithms also have two iteration counters, an outer loop counter (k) and an inner-loop counter (n). The upper bounds for these counters, K and $N(k)$, respectively, define the algorithm's stopping

criteria. When the stopping criterion of an inner loop is met (i.e., $n = N(k)$), then all inner-loop parameters can change (i.e., R_k and $N(k)$). When the stopping criteria of the outer loop are met (i.e., $k = K$), the algorithm terminates. The GHC algorithm is presented in pseudo-code form using inner and outer loops (in contrast to the single loop form described in Section 1):

GHC Algorithm
Inputs:
Set the outer loop counter bound K and the the inner loop counter bounds $N(k)$, $k = 1,2,\ldots,K$
Define a set of hill-climbing random variables $-\infty \le R_k \le +\infty$, $k = 1,2,\ldots,K$
Set the iteration indices $i = 0$, $k = n = 1$
Generate an initial solution $\omega(0) \in \Omega$, and set $\omega^* \leftarrow \omega(0)$
Repeat while $k \le K$
Repeat while $n \le N(k)$
Generate a solution $\omega \in \eta(\omega(i))$ using solution probability mass function g
Generate an observation R from $R_k(\omega(i), \omega)$
Compute $\delta(\omega(i), \omega) = f(\omega) - f(\omega(i))$
If $R \ge \delta(\omega(i), \omega)$, set $\omega(i+1) \leftarrow \omega$; Else (i.e., $R < \delta(\omega(i), \omega)$), set $\omega(i+1) \leftarrow \omega(i)$
If $f(\omega(i+1)) < f(\omega^*)$, *set* $\omega^* \leftarrow \omega(i+1)$
$n \leftarrow n+1$, $i \leftarrow i+1$
Until $n = N(k)$
$n \leftarrow 1$, $k \leftarrow k+1$
Until $k = K$
Output: Report ω^*

Numerous common heuristics can be defined using the GHC algorithm framework (see Jacobson et al. 1998). Pure local search accepts only neighbors of improving (lower) objective-function value. Pure local search can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega) = 0$, for all $\omega(i)$, $\omega \in \eta(\omega(i))$, $k = 1, 2, \ldots, K$. Simulated annealing (Henderson et al. 2003) accepts neighbors of higher objective-function values with decreasing probability, where $P\{R_k(\omega(i), \omega) \ge \delta(\omega(i), \omega)\} = e^{-\delta(\omega(i),\omega)/t_k}$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, and $k = 1, 2, \ldots, K$, for some simulated-annealing temperature parameter $t_k > 0$, where $\lim_{k\to+\infty} t_k = 0$. Simulated annealing can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega) = -t_k \ln(U)$, for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, and $k = 1, 2, \ldots, K$, where $U = U[0, 1]$. Threshold accepting (Dueck and Scheuer 1990) accepts neighbors with higher objective-function values according to a sequence of deterministic (constant) thresholds, Q_k, $k = 1, 2, \ldots, K$. Threshold accepting can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega) = Q_k$ for all $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, and $k = 1, 2, \ldots, K$. Tabu search (Glover and Laguna 1997) can also be formulated within the GHC framework (see Vaughan and Jacobson 2004). Convergence and performance results for GHC algorithms are presented in Johnson and Jacobson (2002ab), Sullivan and Jacobson (2001), and Jacobson and Yücesan (2004a,b).

To formally define a set of *fundamentally related* discrete optimization problems, a metric between elements in a set of fundamentally related discrete optimization problems is introduced to define a distance measure between the discrete optimization problems in the fundamentally related set. This metric aids in designing the problem probability mass function (that governs movement between discrete optimization problems). For example, it is generally desirable to define the SGHC algorithm such that movement between discrete optimization problems with significant similarity occurs more frequently than does movement between discrete optimization problems that are less similar (hence benefit less from an exchange of information). Finally, the SGHC algorithm framework is introduced simultaneously to approach sets of fundamentally related discrete optimization problems using heuristics.

Several definitions are needed to discuss the class of sets of discrete optimization problems on which SGHC algorithms can be applied. Consider a set of discrete optimization problems $S = \{D_1, D_2, \ldots, D_m\}$; problem $D_y = (\Omega_y, f_y)$ is defined by a finite set of solutions Ω_y and a real-valued objective function f_y: $\Omega_y \to \Re$. A set of discrete optimization problems S is *fundamentally related* by a set of objects, $Ob = \{c_1, c_2, \ldots, c_r\}$, if the solution space Ω_y of each problem $D_y \in S$ can be uniquely defined by C_y, a subset of Ob termed the *fundamental relation set* of D_y (i.e., for every problem D_y, there is exactly one set $C_y \subseteq Ob$ such that C_y completely defines Ω_y). For example, in the manufacturing-design problem, the set of objects Ob is the set of all possible manufacturing processes, while for a set of TSPs, the set of objects is the set of all the cities, for a set of Max 3-Satisfiability problems, the set of objects is the set of all the clauses, and for a set of permutation flow-shop problems, the set of objects is the set of all the machines.

Every set of fundamentally related discrete optimization problems can be defined by a set of binary vectors that allow a metric between discrete-optimization problems to be defined. To see this, consider $D_y \in S$ where $C_y \subseteq Ob$ is the fundamental relation set of D_y. Then C_y can be represented by the *binary activity*

vector $\mathbf{c}^y \in \{0,1\}^r$, $\mathbf{c}^y = (\mathbf{c}_1^y, \mathbf{c}_2^y, \ldots, \mathbf{c}_r^y)$, where

$$\mathbf{c}_i^y = \begin{cases} 1 & \text{if } c_i \text{ is contained in } C_y \\ 0 & \text{otherwise} \end{cases}.$$

For the manufacturing-design problem, $\mathbf{c}^1 = \{1, 0, 1, 0, 1, 0, 1\}$, $\mathbf{c}^2 = \{1, 0, 1, 0, 0, 1, 1\}$, $\mathbf{c}^3 = \{1, 0, 1, 1, 1, 0, 1\}$, $\mathbf{c}^4 = \{1, 1, 1, 0, 1, 0, 1\}$, and $\mathbf{c}^5 = \{1, 1, 0, 1, 1, 0, 1\}$.

For two discrete-optimization problems, D_x, $D_y \in S$, if $|C_x \cap C_y| / |C_x \cup C_y|$ is close to one, then the optimal/near-optimal solutions of D_x and D_y may be similar. The following detachment metric is defined to measure if two discrete optimization problems (in a set of fundamentally related discrete optimization problems) are close together. Define the *detachment metric* ρ between discrete optimization problems D_x, $D_y \in S$,

$$\rho(D_x, D_y) = \left|\mathbf{c}_1^x - \mathbf{c}_1^y\right| + \left|\mathbf{c}_2^x - \mathbf{c}_2^y\right| + \ldots + \left|\mathbf{c}_n^x - \mathbf{c}_n^y\right|$$

(see Royden 1988). The detachment metric quantifies overlap between problems in S.

SGHC algorithms seek an optimal solution from among a set of fundamentally related discrete-optimization problems by allowing the algorithm to move probabilistically between such problems. When a new problem is generated, an initial solution for this new problem is then generated using information from the previous problem's best-to-date solution. An inner and outer loop structure is used in SGHC algorithms, where SGHC algorithms restrict possible movement between problems to the first iteration of the outer-loop iterations. This restriction ensures that the embedded GHC algorithm (i.e., the underlying heuristic) is applied to each problem at least $N(k)$ iterations each time it is generated (i.e., initially visited). An SGHC algorithm moves between problems according to a *problem probability mass function* $h_{xy}(k, \rho(D_x, D_y))$, where for all $k = 1,2, \ldots,K$, $0 < h_{xy}(k, \rho(D_x, D_y)) < 1$ for all $D_x \in S$, $D_y \in \eta_{set}(D_x)$, and $\sum_{D_q \in \eta_{set}(D_y)} h_{xy}(k, \rho(D_x, D_y)) = 1$ for all $D_x \in S$, $D_y \in \eta_{set}(D_x)$, where $\eta_{set}(D_x) \subseteq S$ is the set of neighborhood problems for D_x. The two-tuple (k, n) denotes inner-loop iteration $n = 1,2,\ldots, N(k)$; during outer loop iteration $k = 1, 2, \ldots, K$. $D(k)$ is the discrete optimization problem over which the SGHC algorithm is executed during the k^{th} outer-loop iteration, where $\Omega(k)$ is the solution space of $D(k)$. During the inner-loop iterations, the SGHC algorithm is executing over the solution space of the current problem using the embedded GHC algorithm. When the SGHC algorithm terminates, based on the number of inner- and outer-loop iterations executed, the $|S| = m$ best to-date solutions $\omega^*(D)$, $D \in S$, are reported. The SGHC algorithm is presented in pseudo-code form:

SGHC Algorithm

Inputs:
Set the outer loop counter bound K and set the inner loop counter bounds $N(k)$, $k = 1, 2, \ldots, K$
Define a set of hill-climbing random variables $-\infty \le R_k \le +\infty$, $k = 1, 2, \ldots, K$
Select an initial discrete optimization problem $D(0) \in S$ and generate an initial solution $\omega(0) \in \Omega(0)$
Set the iteration indices $N(1) = i = 0$, $k = n = 1$ and set $I(D) = 0$ for all $D \in S$
Set $\omega^*(D(0)) \leftarrow \omega(0)$, $D^* \leftarrow D(0)$ and $I(D(0)) = 1$.
Repeat while $k \le K$
  Generate $D(k) \in \eta_{set}(D(k\text{-}1))$ using the problem probability mass function h
  If $D(k) \ne D(k\text{-}1)$, generate $\omega \in \Omega(k)$ and set $\omega(i) \leftarrow \omega$.
    If $I(D(k)) \ne 1$, set $\omega^*(D(k)) \leftarrow \omega$ and $I(D(k)) = 1$
  If $D(k) = D(k\text{-}1)$, set $\omega(i) \leftarrow \omega(i\text{-}1)$
  Repeat while $n \le N(k)$
    Generate a solution $\omega \in \eta(\omega(i)) \subset \Omega(k)$ using solution probability mass function g
    Compute $\delta(\omega(i),\omega) = f_{D(k)}(\omega) - f_{D(k)}(\omega(i))$ and generate an observation R from $R_k(\omega(i),\omega)$
    If $R \ge \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega$; if $R < \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega(i)$
    If $f_{D(k)}(\omega(i+1)) < f_{D(k)}(\omega^*(D(k)))$, *set* $\omega^*(D(k)) \leftarrow \omega(i+1)$
    $n \leftarrow n + 1$, $i \leftarrow i + 1$
  Until $n = N(k)$
  $n \leftarrow 1$, $k \leftarrow k + 1$
Until $k = K$
Output: Report $\omega^*(D)$ for all $D \in S$

To illustrate further the scope of the SGHC algorithm framework, three illustrative examples of sets of fundamentally related discrete optimization problems are presented: a set of TSPs, a set of Max 3-Satisfiability problems, and a set of permutation flow-shop problems.

The TSP is a well-known NP-hard discrete optimization problem that is useful for modeling a wide variety of real-world problems (Lawler et al. 1985). For example, traditional applications of the TSP include

various types of vehicle routing and scheduling problems. More recently, applications of the TSP have been expanded to include applications like printing circuit boards, x-ray crystallography, overhauling gas turbine engines, and controlling industrial robots (Johnson and Jacobson 2002ab). Informally, the TSP states that, given a set of q cities, we are to find a route that visits all q cities exactly once, returns to the initial city (we have a Hamiltonian cycle), and minimizes the total distance traveled. The TSP is formally stated as follows.

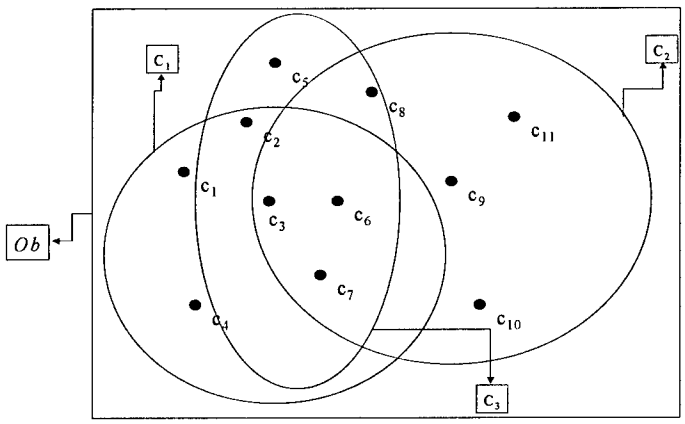
Traveling Salesman Problem:
Instance: Given a set of r cities $C = \{c_1, c_2, \ldots, c_r\}$ and a distance matrix P that represents the cost of traveling between the cities in the set C.
Question: Find a Hamiltonian cycle $h = (c_{(1)}, c_{(2)}, \ldots, c_{(r)})$ (i.e., a permutation of $(c_1, c_2, \ldots, c_r)$) such that $f(h) = \sum_{j=1}^{r-1} P(c_{(j)}, c_{(j+1)}) + P(c_{(r)}, c_{(1)})$ is minimized.

A set of TSPs is defined as follows. Given a set of r cities $Ob = \{c_1, c_2, \ldots, c_r\}$, consider m TSPs, $S = \{D_1, D_2, \ldots, D_m\}$, where each problem D_j is defined by a set of cities C_j such that for all $i = 1, 2, \ldots, r$, if $c_i \in C_j$ then $c_i \in Ob$ (see Figure 1 with $r = 11$ and $m = 3$). The objective is to find the minimal-distance Hamiltonian cycle for each of the m sets of cities.

The set of TSPs is formally stated as follows.

Set of Traveling Salesman Problems:
Instance: Given a set of r cities $Ob = \{c_1, c_2, \ldots, c_r\}$, a set of m subsets of Ob, $O = \{C_1, C_2, \ldots, C_m\}$, and a distance matrix P that represents the cost of traveling between the cities in the set Ob.
Question: Find the m Hamiltonian cycles $h_j = (c_{(1)}, c_{(2)}, \ldots, c_{(|C_j|)}), j = 1,2,\ldots,m$, where $c_{(i)} \in C_j$ for all $i = 1, 2, \ldots, |C_j|$, such that $f_j(h_j) = \Sigma_{y=1,2,\ldots,|C_j|-1} P(c_{(y)}, c_{(y+1)}) + P(c_{(|C_j|)}, c_{(1)}), j = 1, 2, \ldots, m$ are minimized.

Figure 1: Example of *Ob* with $r = 11$ and $m = 3$



Jacobson et al. (2006b) use a set of fundamentally TSPs to model a combat search-and-rescue operation problem for determining which fleet platforms can best search a given area. This resulted in a set of six TSPs. Computational results presented in Jacobson et al. (2006b) demonstrate that near-optimal solutions can be reached more effectively and efficiently using a SGHC algorithm than approaching the problems individually using a GHC algorithm.

Satisfiability was the first problem proven to be NP-complete (Cook 1971). Satisfiability has numerous real-world applications (Gu 1997). For example, the machine-shop scheduling problem consists of a number of operations to be scheduled subject to a collection of constraints, where each operation requires a specified processing time. Smith and Cheng (1993) model this problem with Satisfiability by categorizing the constraints as sequencing restrictions (i.e., an operation must finish before another operation starts), resource capacity constraints (i.e., two operations require the same source, hence cannot be scheduled concurrently), ready times (i.e., the earliest time at which an operation can start), and deadlines (i.e., the latest time at which an operation must be completed). The machine-shop scheduling problem then asks if all operations can be completed on time, without violating any of the constraints.

To describe Satisfiability, several definitions are needed. A *clause* is a combination of *literals,* where a literal is a Boolean variable ($X_i = 1$) or its negation ($X_i = 0$). The *solution space* $\Omega = \{0, 1\}^q$ is the set of all possible solutions (i.e., $|\Omega| = 2^q$), where q is the number of Boolean variables. A solution, $\omega \in \Omega$, is a Boolean vector of size q. Given a solution $\omega \in \Omega$, a clause is satisfied if at least one of its literals takes on the value one.

An instance of Satisfiability consists of a set of y clauses. If there exists (does not exist) an $\omega \in \Omega$ such

that all the clauses are satisfied, then the answer to this instance of Satisfiability is yes (no) and this instance is said to be *satisfiable* (*unsatisfiable*). Several variations of Satisfiability have been studied, based on the number of literals assigned to each clause. For example, if the number of literals is two or three for all clauses, then Satisfiability is referred to as 2-Satisfiability or 3-Satisfiability, respectively. Unless otherwise noted, assume that all clauses contain three literals.

Any 3-Satisfiability problem can be formulated as an optimization problem (termed Max 3-Satisfiability) by introducing the objective function $f = \Sigma_{j=1,2,\ldots,y} P_j(\omega)$, where the goal is to maximize f over the solution space Ω, where

$$P_j(\omega) = \begin{cases} 1 & \text{if clause } j \text{ is satisfied for solution } \omega \\ 0 & \text{otherwise} \end{cases}.$$

3-Satisfiability (Max 3-Satisfiability) is formally defined as follows.

3-Satisfiability (Max 3-Satisfiability):
Instance: Given a collection of y clauses defined by q literals, where each clause contains three literals.
Question: Find a solution ω (i.e., a set of values for the q Boolean variables) such that $f = \Sigma_{j=1,2,\ldots,y} P_j(\omega) / y$ is equal to one (maximized).

A set of 3-Satisfiability (Max 3-Satisfiability) problems is defined as follows. Given a set of r clauses $Ob = \{c_1, c_2, \ldots, c_r\}$ and q literals, consider m 3-Satisfiability (Max 3-Satisfiability) problems $S = \{D_1, D_2, \ldots, D_m\}$, where each D_j is defined by a set of clauses C_j, and for each $c_i \in C_j$, $c_i \in Ob$. For each of these m problems, find a set of values for the q Boolean variables such that f_j (the objective function for D_j) is equal to one (maximized).

Set of 3-Satisfiability (Max 3-Satisfiability) Problems:
Instance: Given a set of r satisfiability clauses $Ob = \{c_1, c_2, \ldots, c_r\}$ and a set of m subsets of Ob, $O = \{C_1, C_2, \ldots, C_m\}$, where each clause contains three literals..
Question: For each C_i, $i = 1, 2, \ldots, m$, find a solution ω_i (i.e., a set of values for the q Boolean variables) such that $f_i = (\Sigma_{j=1,2,\ldots,|C_i|} P_j(\omega_i)) / |C_i|$, is equal to one (maximized).

An example of when such a set of discrete optimization problems may occur is for a machine-shop scheduling problem that considers similar, yet different, sets of constraints. In particular, consider a machine-shop scheduling problem defined by a given set of constraints. A second (fundamentally related) machine-shop scheduling problem may be under consideration defined by the same set of constraints slightly perturbed (e.g., optional deadlines, deadlines that fluctuate according to the day of the week, amd processing times that vary).

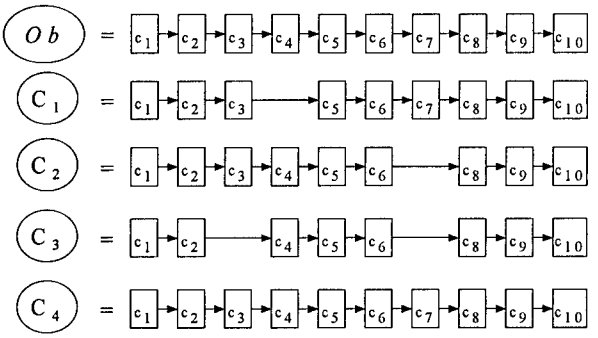
Scheduling problems have a broad spectrum of application domains (e.g., manufacturing, computers, and health care). A set of permutation flow-shop problems is used to further illustrate the concept of a fundamentally related set of discrete optimization problems. Informally, given a set of q jobs to be processed on r machines (in the order 1, 2, ..., r), the permutation flow-shop problem seeks to find a sequence of jobs that minimize the maximum completion time C_{max}, where preemption is not permitted (Aarts and Lenstra 1997). For three or more machines, the permutation flow-shop problem is NP-hard. The permutation flow-shop problem is formally stated as follows.

Permutation Flow-Shop Problem:
Instance: Given a set of r machines $C=\{c_1, \ldots, c_r\}$, a set of q jobs $J=\{j_1, \ldots, j_q\}$ to be processed on the r machines (in the order 1,2,..., r), and a matrix P (where p_{ij} denotes the processing time of job i on machine j).
Question: Find a sequence of jobs such that C_{max}, the maximum completion time, is minimized.

A set of permutation flow-shop problems is defined as follows. Given a set of r machines $Ob = \{c_1, c_2, \ldots, c_r\}$, consider m permutation flow-shop problems $S = \{D_1, D_2, \ldots, D_m\}$, where each D_j is defined by a set of machines C_j such that for all $i = 1, 2, \ldots, r$, if $c_i \in C_j$ then $c_i \in Ob$ (see Figure 2 with $r = 10$ and $m = 4$). The same q jobs are to be considered for each set C_i. For each of these m problems, the objective is to find a sequence of jobs that minimizes the maximum completion time C_{max}. The set of permutation flow-shop problems is formally stated as follows.

Set of Permutation Flow-Shop Problems:
Instance: Given a set of r machines $Ob = \{c_1, c_2, \ldots, c_r\}$, a set of m subsets of Ob, $O = \{C_1, C_2, \ldots, C_m\}$, where for each $C_w \in O$, $w = 1, 2, \ldots, m$, the set of q jobs $J = \{j_1, j_2, \ldots, j_q\}$ is to be processed on $|C_w|$ machines (in ascending order) and a matrix P (where $p_{i1,j2}$ denotes the processing time of job $i1$ on machine $j2$).
Question: Find a sequence of jobs for each set of machines C_w, $w = 1, 2, \ldots, m$, such that C_{max}, the maximum completion time, is minimized.

Figure 2: Set of Permutation Flow-Shop Problems, with $r = 10$ and $m = 4$



The previous discussion also suggests how to model many variations of the classical job-shop scheduling problem (Aarts and Lenstra 1997). The classical job-shop scheduling problem is very similar to the permutation flow-shop problem without the restriction that the machines are visited in a predetermined order. The job-shop scheduling problem approximates a manufacturing process. However, in a situation in which there is only one of each type of machine, certain machines may become bottlenecks or perhaps fail. Therefore, in practice, machines requiring high availability or disproportionately large processing times are very likely replicated (Barnes and Chambers 1995). The flexible job-shop problem (FJSP) is an extension of the classical job-shop scheduling problem. In the FJSP, any one machine from a given set of machines can process an operation, where the problem is to assign each operation to a machine and to order the operations on the machines such that the maximal completion time of all operations is minimized.

Sets of fundamentally related job-shop scheduling problems occur when several manufacturing-process options are being considered. For example, consider a set of processes to develop an airplane part. To enhance the part, inserting an additional inspection process or a process that adds or adjust seals may be under consideration. Similarly, to reduce costs, eliminating an existing process may be desirable. Moreover, sets of fundamentally related flexible job-shop problems occur in practice, such as when bottlenecks are found, they can often be eliminated by adding one or more machines to some of the sets of machines. Exploring the options of machine additions (and which machine to purchase: *"should we buy a new machine A or machine B?"*) generates a set of fundamentally related discrete-optimization problems.

Note that when the SGHC algorithm moves between discrete-optimization problems, information gained while optimizing over the current discrete-optimization problem D_i is used to obtain the initial solution in the subsequent discrete-optimization problem D_j. For the set of TSPs addressed in Jacobson et al. (2006b) and Vaughan et al. (2005), this is accomplished by first recording the best solution to date found while optimizing over D_i. Then, for every pair of cities contained in this solution, if this pair is a feasible city pair for D_j, then it is incorporated into the initial solution for D_j. Note that not every edge in this solution for D_i is likely to be incorporated into the initial solution for D_j. However, once all possible edges are passed, the remaining edges can then be randomly generated. Note that an alternate approach to complete the initial solution for D_j would be to generate the remaining edges by applying a greedy algorithm. Lastly, for both the set of Max 3-Satisfiability problems and the set of permutation flow-shop problems, the initial solution for the subsequent problem can be obtained by recording the best solution (Boolean vector, job sequence) found to date, while optimizing over discrete-optimization problem D_i and using this solution (Boolean vector, job sequence) as the initial solution for the subsequent discrete-optimization problem D_j.

Vaughan et al. (2005) report new computational results using the SGHC algorithm to address a randomly generated set of four TSPs and a randomly generated set of four Max 3-Satisfiability problems. For the set of four traveling-salesman (Max 3-Satisfiability) problems, simulated annealing (threshold accepting) is embedded in the SGHC algorithm framework. For comparison purposes, the same heuristics are also applied individually to each discrete-optimization problem in the sets. A total of 30 independent replications were executed for each SGHC and GHC algorithm formulation, where each replication was initialized with a different randomly generated initial solution. All the experiments were run in Matlab 6.5 on a 2.4MHz Pentium IV with 512 MB of RAM. The means μ and the standard deviations σ were computed from the objective-function values of the best solutions found for each of the 30 replications. These measures allow the SGHC and the GHC algorithms to be compared computationally, and hence assess their relative effectiveness and efficiency.

Each set of problems addressed with the SGHC algorithms used the problem probability mass function

$$h_{D_xD_y}(k, \rho(D_x, D_y)) = [1 / \rho(D_x, D_y)] / [\sum_{\substack{i=1 \\ i \neq x}}^{4} (1 / \rho(D_x, D_y))], x \neq y \quad (15)$$

$$h_{D_xD_x}(k, \rho(D_x, D_x)) = 1 - \sum_{\substack{q=1 \\ q \neq x}}^{4} [1 / (\rho(D_x, D_q)] / [\sum_{\substack{i=1 \\ i \neq x}}^{4} (1 / \rho(D_x, D_i))] = 0, \quad (16)$$

for every $y,q = 1, 2, 3, 4$, $y \neq q$ and for every $k = 1, 2, \ldots, K$, with the detachment metrics $\rho(D_y, D_q)$ reported in Vaughan et al. (2005). This problem probability mass function results in a stationary and ergodic Markov chain $\{\Psi(k)\}$ (see Vaughan et al. 2005) with transition matrix $T(k)$, where $T_{xy}(k) = h_{xy}(k, \rho(D_x, D_y))$. Therefore, if $T = T(k)$ is irreducible and aperiodic, then as k approaches infinity, the SGHC algorithm is executing over the solution space of each discrete optimization problem in S with probability π, where $\pi = \pi T$ and $\sum_{i=1}^{m} \pi_i = 1$ (Isaacson and Madsen 1985). Moreover, this problem probability mass function guarantees that the discrete optimization problem over which the SGHC algorithm is executing changes at every outer-loop iteration k (i.e., $\Psi(k) \neq \Psi(k\text{-}1)$, for all $k = 1, 2, \ldots$).

To illustrate the application of an SGHC algorithm on a set of TSPs, 25 cities were randomly generated on a 100 × 100 unit grid. Four TSPs (D_1, D_2, D_3, D_4) of size $|\Omega_y| = 20$, 22, 24, and 25, for $y = 1, 2, 3, 4$, were created by randomly selecting $|\Omega_y|$ of the 25 cities for each TSP.

Computational results with simulated-annealing SGHC algorithms are reported. For comparison purposes, computational results with simulated annealing applied to each problem individually are also reported. The 2-opt neighborhood function (Aarts and Lenstra 1997) was used for all executions of the SGHC and GHC algorithms. The SGHC detachment metric was used in (15) and (16) to create the transition matrix

$$T = \begin{bmatrix} 0 & 0.2703 & 0.4054 & 0.3243 \\ 0.1667 & 0 & 0.5000 & 0.3333 \\ 0.1429 & 0.2857 & 0 & 0.5714 \\ 0.1304 & 0.2174 & 0.6522 & 0 \end{bmatrix}, \quad (17)$$

with stationary distribution $\pi = (.1259, .2041, .3571, .3129)$. The fraction of iterations that the SGHC algorithm actually searched in each problem is $\hat{\pi} = (.122, .204, .360, 0.317)$, which is an estimator for π. The SGHC algorithm inner- and outer-loop bounds were $K = 400$ and $N = N(k) = 400$ for all k. The inner- and outer-loop bounds for the four GHC algorithms were $K = 250$ and $N = N(k) = 160$ for all k. Therefore, the total number of iterations executed (i.e., 160,000) for the single SGHC algorithm execution and for the four GHC algorithm executions (one for each discrete optimization problem) was identical.

For simulated annealing, t_k was updated by multiplying the previous temperature parameter by the increment multiplier $\beta = 0.98$ (i.e., $t_k = \beta t_{k-1}$) with initial temperature parameter $t_0 = (0.2)(25)(M)$, where M is the maximum distance between any two of the 25 cities. The hill-climbing random variable was $R_k(\omega(i), \omega) = -t_k \ln(U)$, for all $\omega(i)$, $\omega \in \eta(\omega(i))$, and $k = 1, 2, \ldots, K$, where $U = \mathrm{U}(0, 1)$. These values were found by determining (experimentally) the parameters that produced the best results for the simulated-annealing algorithm applied to the four problems. Note that the values reported for μ and σ in Tables 1-3 are rounded to one decimal.

Table 1 Simulated Annealing GHC and SGHC Algorithm Results

Algorithm		D_1	D_2	D_3	D_4
GHC	μ	407.0	426.1	436.3	441.7
	σ	2.3	5.0	6.9	6.6
SGHC	μ	405.7	422.9	433.2	443.7
	σ	0.5	1.9	3.5	6.5

The results in Table 1 suggest that the SGHC algorithm slightly outperformed the GHC algorithm (as measured by μ.) for the first three problems being addressed, and slightly underperformed the last problem (which also corresponds to the problem with the largest objective function value). The SGHC algorithm results also resulted in smaller standard deviations than the results obtained with the GHC algorithms, which indicate that the SGHC algorithm results are more predictable. Note that the SGHC and GHC algorithms results took approximately the same time to execute (the average CPU time per set of thirty replications was 670 seconds for the SGHC algorithm and 169 seconds for each GHC algorithm).

The best solutions found for the four problems D_1, D_2, D_3 and D_4 were 405.6, 421.6, 430.4, and 435.8 respectively. The SGHC algorithm found these solutions 28, 16, 5, and 3 times (out of the thirty replications), respectively, while the GHC algorithms found these solutions 18, 7, 3, and 4 times (out of the thirty replications), respectively. This suggests that for the problem with the overall best solution, the SGHC algorithm may be most effective in finding the corresponding best solution.

To assess further the effectiveness of SGHC algorithm, a sequential version of GHC algorithms (termed sequential GHC) was applied to the four problems, D_1, D_2, D_3, D_4. For example, for the sequence of problems $D_1D_2D_3D_4$, simulated annealing was applied to problem D_1 with $K = 250$ and $N = N(k) = 160$ for all k. Then, the best solution obtained for problem D_1 was used to create an initial solution for simulated annealing applied to problem D_2, with $K = 250$ and $N = N(k) = 160$ for all k. The same procedure was then followed (in sequence) for problems D_3 and D_4. Note that all 4! = 24 permutations of D_1, D_2, D_3, D_4 were considered, with the results for all these experiments reported in Table 2. The SGHC algorithm results were slightly better than the best results obtained using this sequential GHC algorithm over the 24 permutations of D_1, D_2, D_3, D_4 (see the bold values in Table 2). Moreover, the CPU time (averaged over the 24 permutations of D_1, D_2, D_3 and D_4) per set of thirty replications was 668 seconds. This suggests that the SGHC algorithm may be an efficient means to get the benefit of considering all problem permutations for the sequential application of GHC algorithms.

To assess the impact of the detachment-metric approach in defining the transition matrix, the following two uniform transition matrices were considered:

$$T_1 = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \tag{18}$$

and

$$T_2 = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}, \tag{19}$$

both with stationary distributions π = (0.25, 0.25, 0.25, 0.25). Transition matrix T_1 results in random movement across the four problems (including the current problem), while matrix T_2 results in random movement to a new problem, at each outer-loop iteration. The SGHC algorithm results using these two transition matrices are reported in Table 3. Comparing these results with Table 1 suggest that the SGHC algorithms yielded almost identical results across the three different transition matrices. The average computational time using T_1 and T_2 were comparable to those obtained with T. Therefore, the primary benefit of using the detachment metric approach in defining the transition matrix may be to satisfy the convergence conditions in Vaughan et al. (2005). This would be analogous to designing simulated-annealing algorithms based on convergence conditions that are rarely used in practice. Further research is needed to assess better the impact of the transition matrix on the performance of the SGHC algorithm.

A set of four Max 3-Satisfiability problems was obtained by randomly generating 90 clauses from 20 literals, where each literal is negated with probability ½. Four Max 3-Satisfiability problems (D_1, D_2, D_3, D_4) of sizes $|\Omega_y|$ = 84, 85, 86, and 90, for y = 1, 2, 3, 4, were defined by randomly selecting $|\Omega_y|$ of the 90 clauses for each Max 3-Satisfiability problem. These four problems have clause-to-variable ratios of 4.2, 4.25, 4.3, and 4.5, respectively. Note that the hardest Max 3-Satisfiability problems occur when the clause-to-variable ratio is approximately 4.25 (see Selman et al. 1992).

Computational results with threshold-accepting SGHC algorithms are reported. To be consistent with the set of TSP, the Max 3-Satisfiability problems are formulated as minimization problems (e.g., 95.7% of the clauses satisfied is expressed as an objective function value of -0.957). Computational results are also reported with threshold-accepting applied to each problem individually. The values reported for μ and σ in Tables 4-6 are rounded to three decimal places.

Using the detachment metric described in Vaughan et al. (2005), the resulting transition matrix is

$$T = \begin{bmatrix} 0 & 0.2376 & 0.3267 & 0.4356 \\ 0.2261 & 0 & 0.2764 & 0.4975 \\ 0.2571 & 0.2286 & 0 & 0.5143 \\ 0.2703 & 0.3243 & 0.4054 & 0 \end{bmatrix},$$

with stationary distribution $\pi = (.2027, .2130, .2576, .3267)$. The actual fraction of iterations that the SGHC algorithm searched in each problem is $\hat{\pi} = (.\ 203, .210, .261, .327)$. The SGHC algorithm inner- and outer-loop bounds were $K = 400$ and $N = N(k) = 3$ for all k. The inner- and outer-loop bounds for the four GHC algorithms were $K = 300$ and $N = N(k) = 1$ for all k. Therefore, the total number of iterations executed (i.e., 1,200) for the single SGHC algorithm execution and the four GHC algorithm executions (one for each problem) were identical. For threshold accepting, the hill-climbing random variable is $R_k(\omega(i), \omega) = Q_k$, for all $\omega(i)$, $\omega \in \eta(\omega(i))$, and $k = 1, 2, \ldots, K$, where $Q_k = (0.98)^k Q_0$ with $Q_0 = 1$.

Table 2 Sequential Simulated Annealing GHC Algorithm Results

	D_1		D_2		D_3		D_4	
Sequence	μ	σ	μ	σ	μ	σ	μ	σ
$D_1D_2D_3D_4$	407.5	2.4	425.6	4.5	436.5	6.0	441.3	5.6
$D_1D_2D_4D_3$	406.7	1.6	425.6	4.9	434.8	5.9	442.4	5.8
$D_1D_3D_2D_4$	407.4	1.7	425.7	5.2	434.5	5.4	444.1	7.1
$D_1D_3D_4D_2$	406.9	1.3	425.8	4.0	435.5	7.1	442.9	7.8
$D_1D_4D_3D_2$	406.7	1.6	424.8	3.9	435.5	5.9	441.2	6.7
$D_1D_4D_2D_3$	406.8	1.6	424.7	4.5	436.6	7.3	442.5	6.6
$D_2D_1D_3D_4$	406.7	1.5	424.7	4.1	**432.5**	3.9	440.9	5.3
$D_2D_1D_4D_3$	407.1	2.0	425.2	5.2	435.9	6.9	442.2	6.7
$D_2D_3D_1D_4$	406.7	1.7	425.2	5.4	434.3	5.9	441.6	5.8
$D_2D_3D_4D_1$	406.6	1.9	**423.3**	2.2	435.9	7.0	442.3	7.5
$D_2D_4D_1D_3$	407.0	1.5	425.8	4.8	437.3	7.3	441.8	6.5
$D_2D_4D_3D_1$	406.7	1.4	426.0	5.7	434.4	6.0	441.6	7.0
$D_3D_1D_2D_4$	**406.3**	1.4	424.7	4.4	435.9	6.5	443.2	6.9
$D_3D_1D_4D_2$	407.0	1.4	425.4	4.2	434.9	5.4	441.4	6.1
$D_3D_2D_1D_4$	407.4	2.7	423.4	2.0	435.0	5.6	443.3	7.0
$D_3D_2D_4D_1$	406.6	1.3	425.0	4.8	436.2	6.5	445.5	8.0
$D_3D_4D_1D_2$	407.0	2.0	424.6	3.9	434.1	5.1	443.7	7.6
$D_3D_4D_2D_1$	407.2	2.6	425.2	5.1	434.3	5.2	441.7	7.1
$D_4D_1D_2D_3$	406.8	1.6	424.8	3.1	433.5	4.1	443.1	7.0
$D_4D_1D_3D_2$	406.9	1.6	424.9	5.6	435.5	5.5	**439.6**	6.0
$D_4D_2D_1D_3$	407.3	1.6	425.8	5.6	435.5	7.3	440.7	4.5
$D_4D_2D_3D_1$	406.5	1.8	424.8	4.5	435.2	6.6	442.4	7.2
$D_4D_3D_1D_2$	407.1	1.9	425.6	5.6	434.3	4.6	441.6	6.8
$D_4D_3D_2D_1$	406.7	1.4	423.4	2.4	434.7	5.6	444.5	7.3

Table 3 Simulated Annealing SGHC Algorithm Results (with T_1 and T_2)

Transition Matrix		D_1	D_2	D_3	D_4
T_1	μ	405.7	423.0	435.7	443.4
	σ	0.5	2.1	5.6	7.3
T_2	μ	405.8	422.6	435.8	443.5
	σ	0.7	1.5	5.5	5.6

Table 4 Threshold-Accepting GHC and SGHC Algorithm Results

Algorithm		D_1	D_2	D_3	D_4
GHC	μ	-0.988	-0.993	-0.988	-0.986
	σ	0.009	0.009	0.011	0.012
SGHC	μ	-0.994	-0.995	-0.994	-0.993
	σ	0.006	0.007	0.007	0.007

The results in Table 4 suggest that the SGHC algorithm is more effective than the GHC algorithm, as measured by μ, for the problem being studied. Moreover, over the 30 replications, the SGHC algorithm was able to determine that each of the four problems were satisfiable for 16, 19, 16 and 14 replications, respectively, while the GHC algorithms were able to determine that each of the four problems were satisfiable for 8, 18, 8 and 8 replications, respectively. In addition, the standard deviations for the SGHC algorithm are significantly smaller than the standard deviations for the GHC algorithm. The CPU time per set of thirty replications was 96 seconds for the SGHC algorithm and 169 seconds (on average) for each GHC algorithm.

As described for the TSPs, a sequential application of GHC algorithms was applied to D_1, D_2, D_3, and D_4. In particular, threshold-accepting was applied to problems D_1, D_2, D_3, and D_4 in sequence, with $K = 300$ and $N = N(k) = 1$ for all k, where the best solution obtained for the current problem was used to create an initial solution for threshold-accepting applied to the subsequent problem in sequence. As for the TSPs, all $4! = 24$ permutations of D_1, D_2, D_3, D_4 were considered, with the results over all these experiments reported in Table 5. Once again, the SGHC algorithm results were comparable to the best results (see the bold values in Table 5) obtained using this sequential GHC algorithm. The CPU time (averaged over the 24 permutations of D_1, D_2, D_3 and D_4) per set of thirty replications was 165 seconds.

To assess once again the effectiveness of the detachment-metric approach in defining the transition matrix, transition matrices T_1 and T_2 were considered. The results with the SGHC algorithm using these two transition matrices are reported in Table 6. The average computation time using T_1 and T_2 were comparable to those obtained with transition matrix T. The same observations that were made for T, T_1, and T_2, used for the set of TSPs also apply for the set of Max 3-Satisfiability problems.

The SGHC algorithm framework has been introduced to address sets of related discrete optimization problems, that generalizes the results reported in Vaughan et al. (2000) for a set of manufacturing-design problems. Jacobson et al. (2006b) also report the application of this framework for a set of combat search-and-rescue operation problems.

4. Neighborhood Function Design Properties

An outcome of this research effort has been an extensive analysis to gain a more thorough understanding of the properties of neighborhood functions for local search algorithms applied to hard discrete optimization problems. Armstrong and Jacobson (2005) consider the requirements of neighborhood functions for local search algorithms that ensure that such algorithms can find global optima. They prove that data independent neighborhood functions with the smooth property (i.e., all strict local optima are global optima) for Max 3-Satisfiability must contain all possible solutions for large instances, that data independent neighborhood functions with the smooth property for 0-1 Knapsack are shown to have size with the same order of magnitude as the cardinality of the solution space, and that data independent neighborhood functions with the smooth property for TSP are shown to have exponential size. These results also hold for certain polynomially solvable sub-problems of Max 3-Satisfiability, 0-1 Knapsack and Traveling Salesman Problem (TSP).

The effectiveness of local search algorithms (Papadimitriou and Steiglitz 1982) on discrete optimization problems is highly dependent on the choice of neighborhood function. The results describe here prove that the only data independent neighborhood functions with the smooth property (all strict local optima are global optima) for Maximum 3-Satisfiability (Garey and Johnson 1979) are neighborhood functions that contain all possible solutions for large instances. More precisely, if a given neighborhood function for Maximum 3-Satisfiability has the smooth property, then, for instances with $n \geq 4$ Boolean variables, the neighborhood function of every solution x contains all possible solutions except for the solution x itself. A result for 0-1 Knapsack shows that data independent neighborhood functions with the smooth property must have size that is $\Theta(2^n)$ where n denotes the number of items in the problem instance. Furthermore, a neighborhood function η^K with the smooth property for 0-1 Knapsack is given so that if $\eta(I, x) \subset \eta^K(I, x)$ for some instance I and solution x, then η does not have the smooth property. The neighborhood function η^K is said to be the minimal data independent neighborhood function with the smooth property for 0-1 Knapsack. Note that a neighborhood function η^{MS} consisting of all solutions for instances of Maximum 3-Satisfiability (with $n \geq 4$ Boolean variables) also has the property that if $\eta(I, x) \subset \eta^{MS}(I, x)$ (for an instance I with $n \geq 4$ Boolean variables) and solution x, then η does not have the smooth property. The results reported here also show that if a given TSP neighborhood function (Lawler et al. 1985) has the smooth property, then the neighborhood of every solution has cardinality $\Omega(2^{n/3})$, where n denotes the number of cities in the problem instance.

The results described here are obtained by constructing instances of the discrete optimization problem such that specified data independent neighborhood functions have a strict local optimum that is not a global optimum. In particular, instances are created where there is a unique global optimum and a unique solution with the second best objective function value. The solution with the second best objective function value is chosen such that the unique global optimum is not in its neighborhood. This implies that the solution with the second best objective function value is a strict local optimum. By construction, the classes of instances used in the proofs form polynomially solvable sub-problems of Maximum 3-Satisfiability, 0-1 Knapsack and TSP.

Therefore, the results listed in the first paragraph also hold for polynomially solvable sub-problems of Maximum 3-Satisfiability, 0-1 Knapsack and TSP. For example, there exists a polynomially solvable sub-problem of Maximum 3-Satisfiability such that data independent neighborhood functions with the smooth property must contain all possible solutions for instances with $n \geq 4$ Boolean variables.

Table 5 Sequential Threshold Accepting GHC Algorithm Results

	D_1		D_2		D_3		D_4	
Sequence	μ	σ	μ	σ	μ	σ	μ	σ
$D_1D_2D_3D_4$	-0.991	0.009	-0.993	0.006	-0.991	0.008	-0.991	0.007
$D_1D_2D_4D_3$	-0.991	0.009	-0.991	0.008	-0.993	0.006	-0.989	0.009
$D_1D_3D_2D_4$	-0.991	0.009	**-0.995**	0.007	-0.992	0.007	-0.990	0.010
$D_1D_3D_4D_2$	-0.986	0.010	-0.993	0.010	-0.991	0.007	-0.992	0.007
$D_1D_4D_3D_2$	-0.992	0.006	-0.993	0.009	-0.993	0.008	-0.988	0.010
$D_1D_4D_2D_3$	-0.990	0.007	-0.993	0.008	-0.991	0.010	**-0.994**	0.007
$D_2D_1D_3D_4$	**-0.993**	0.009	-0.994	0.007	**-0.997**	0.005	-0.990	0.007
$D_2D_1D_4D_3$	-0.991	0.009	-0.992	0.008	-0.990	0.008	-0.992	0.006
$D_2D_3D_1D_4$	-0.989	0.010	-0.991	0.010	-0.993	0.008	-0.990	0.010
$D_2D_3D_4D_1$	-0.992	0.007	**-0.995**	0.007	-0.991	0.006	-0.990	0.009
$D_2D_4D_1D_3$	-0.991	0.008	-0.993	0.009	-0.991	0.006	-0.991	0.008
$D_2D_4D_3D_1$	-0.990	0.008	-0.992	0.008	-0.992	0.007	-0.991	0.007
$D_3D_1D_2D_4$	-0.991	0.009	-0.993	0.008	-0.991	0.007	-0.989	0.009
$D_3D_1D_4D_2$	-0.990	0.006	**-0.995**	0.008	-0.993	0.008	-0.990	0.006
$D_3D_2D_1D_4$	-0.991	0.008	-0.993	0.010	-0.989	0.009	-0.989	0.008
$D_3D_2D_4D_1$	-0.990	0.009	-0.993	0.010	-0.988	0.011	-0.993	0.007
$D_3D_4D_1D_2$	-0.989	0.007	-0.993	0.008	-0.991	0.008	-0.989	0.009
$D_3D_4D_2D_1$	-0.992	0.007	-0.992	0.008	-0.993	0.009	-0.991	0.007
$D_4D_1D_2D_3$	-0.992	0.009	-0.993	0.008	-0.993	0.007	-0.993	0.006
$D_4D_1D_3D_2$	-0.992	0.006	-0.989	0.008	-0.992	0.007	-0.991	0.011
$D_4D_2D_1D_3$	-0.990	0.008	-0.992	0.008	-0.994	0.008	-0.987	0.012
$D_4D_2D_3D_1$	-0.990	0.009	-0.993	0.008	-0.991	0.008	-0.989	0.012
$D_4D_3D_1D_2$	-0.989	0.008	-0.992	0.007	-0.992	0.008	-0.991	0.011
$D_4D_3D_2D_1$	-0.990	0.008	-0.990	0.010	-0.993	0.006	-0.989	0.010

Table 6 Threshold-Accepting SGHC Algorithm Results (with T_1 and T_2)

Transition Matrix		D_1	D_2	D_3	D_4
T_1	μ	-0.991	-0.993	-0.991	-0.991
	σ	0.008	0.009	0.010	0.009
T_2	μ	-0.992	-0.993	-0.992	-0.992
	σ	0.007	0.008	0.008	0.008

A *neighborhood function* for problem Π in NPO (NP Optimization) (Ausiello et al. 1999) is a rule that maps an instance and feasible solution pair (I, x), where $I \in D$ and $\boldsymbol{x} \in \text{SOL}(I)$, to a set of feasible solutions. Therefore, a neighborhood function η for problem Π satisfies $\eta(I, \boldsymbol{x}) \subseteq \text{SOL}(I)$ for every instance $I \in D$ and every solution $\boldsymbol{x} \in \text{SOL}(I)$. Given an instance and feasible solution pair (I, x), where $I \in D$ and $\boldsymbol{x} \in \text{SOL}(I)$, $\eta(I, \boldsymbol{x})$ is referred to as the neighborhood of solution x. Note that a solution is not permitted to be a member of its own neighborhood (i.e., $\boldsymbol{x} \notin \eta(I, \boldsymbol{x})$ for all instances $I \in D$ and solutions $\boldsymbol{x} \in \text{SOL}(I)$). This restriction is consistent and compatible with the local search algorithm formulation.

To characterize properties of neighborhood functions, the following definitions are needed. Define the *size of a neighborhood function* η for an instance I to be $\max_{x \in \text{SOL}(I)} |\eta(I, x)|$. A neighborhood function η for Π is *complete* if $\eta(I, \boldsymbol{x}) = \text{SOL}(I) - \{\boldsymbol{x}\}$ for every instance $I \in D$ (with length[I] sufficiently large, since the size of the neighborhood function is analyzed asymptotically) and $\boldsymbol{x} \in \text{SOL}(I)$. A neighborhood function in which all

local optima are global optima is said to have the *global search* (*GS*) *property*. A neighborhood function in which all strict local optima are global optima is said to have the *smooth property*. For every solution x, a neighborhood function that can be searched in polynomial time for an improving solution or else x is deemed a local optimum is said to be *polynomially computable*.

The following definitions will be given for a maximization problem. A solution $x \in \text{SOL}(I)$ is a (*strict*) *local optimum* if $m(I, x)$ (>) $\geq m(I, y)$ for all $y \in \eta(I, x)$, and a solution $x \in \text{SOL}(I)$ is a *global optimum* if $m(I, x) \geq m(I, y)$ for all $y \in \text{SOL}(I)$. Data independent neighborhood functions are defined for discrete optimization problems in NPO that can be formulated as *consistent* optimization problems (i.e., there exists a sequence of sets $\{\Lambda_n\}_{n=1}^{\infty}$ such that $\Lambda_n \subseteq \{0,1\}^n$ and every instance I can be represented as max $m(I, x)$ subject to $x \in \Lambda_n$, where m is a polynomially computable objective function and n is a positive integer that is polynomial in the length of instance I). To be *independent* of the problem data, a neighborhood function η must satisfy the following property for all positive integers n:
Let I_1 and I_2 be instances denoted as max $m(I_1, x)$ subject to $x \in \Lambda_n$, and max $m(I_2, x)$ subject to $x \in \Lambda_n$, respectively. Then $\eta(I_1, x) = \eta(I_2, x)$ for all $x \in \Lambda_n$.
The data independent neighborhood function definition depends on the representation of the problem as a consistent optimization problem. Therefore, for the optimization problems discussed here, the sets $\{\Lambda_n\}_{n=1}^{\infty}$ will be specified. In particular, Maximum 3-Satisfiability and 0-1 Knapsack are consistent where $\{\Lambda_n\}_{n=1}^{\infty}$ are all Boolean vectors over n dimensions, while TSP is consistent where $\{\Lambda_n\}_{n=1}^{\infty}$ are the collection of distinct Hamiltonian tours over the n cities. A neighborhood function is *reasonable* if it is independent of the problem data and has polynomial size. Reasonable neighborhood functions have been studied since it is conjectured that their properties may indicate the difficulty of a discrete optimization problem (Tovey 1985). Note that the restriction to polynomially sized neighborhood functions is not, in general, always necessary for iterations of a local search algorithm to be completed in polynomial time. In particular, there exist several exponentially large neighborhood functions for NP-hard discrete optimization problems, such as TSP, which can be searched in polynomial time (Ahuja et al. 2003).

A limited number of papers report results that prove that certain discrete optimization problems have no reasonable neighborhood function with the GS or smooth properties. Vizing (1977) and Savage et al. (1976) independently show that any problem parameter independent neighborhood function of TSP for which all local optima are global optima must be exponentially large, hence there does not exist a reasonable neighborhood function for TSP that has the GS property. This result is extended here by showing that data independent neighborhood functions with the smooth property must have size of $\Omega(2^{n/3})$ where n denotes the number of cities in the TSP instance. Papadimitriou and Steiglitz (1978) show that all k-opt neighborhood functions for TSP do not have the GS property and their local optima can have cost that is arbitrarily worse than the cost of global optima. In particular, Papadimitriou and Steiglitz (1978) show that there exist instances of TSP with $8n$ cities, for which there is a unique optimal tour and $2^{n-1}(n-1)!$ tours that are second best with arbitrarily high cost. Furthermore, all of these $2^{n-1}(n-1)!$ tours that are second best cannot be improved without changing fewer than $3n$ edges. Tovey (1985) shows that every reasonable neighborhood function for Maximum Clique and Maximum Satisfiability do not have the smooth property. This Maximum 3-Satisfiability result is strengthened here by showing that all data independent neighborhood functions for Maximum 3-Satisfiability do not have the smooth property, except for neighborhood functions that contain all possible solutions for instances with $n \geq 4$ Boolean variables. Rodl and Tovey (1987) also demonstrate that for Maximum Independent Set, there exists a graph G (up to the relabeling of the vertices) such that all neighborhood functions of polynomial size have exponentially many local optima.

Showing that particular NP-hard discrete optimization problems do not have a reasonable neighborhood function with the smooth property is important since it is conjectured that this condition is characteristic of all NP-hard discrete optimization problems. In other words, it is conjectured that all NP-hard discrete optimization problems have the property that every reasonable neighborhood function does not have the smooth property. This result is likely to be hard to prove in general since it implies that $P \neq NP$ (Tovey 1985). Conversely, a discrete optimization problem is not necessarily hard if it does not have a reasonable neighborhood function with the smooth property, since there exist polynomially solvable such problems that do not have a reasonable neighborhood function with the smooth property. The results reported here for Maximum 3-Satisfiability, 0-1 Knapsack and TSP also hold for corresponding polynomially solvable sub-problems.

The results reported do not rely on complexity theory assumptions; they show that a large number of data independent neighborhood functions for Maximum 3-Satisfiability, 0-1 Knapsack, and TSP do not have the smooth property. This is in contrast to a similar result in Yannakakis (1997) that relies on the assumption that $P \neq NP$ or $NP \neq$co-NP. Suppose that Π is an optimization problem and η is a neighborhood function such that

the local search problem (Π, η) is in PLS (Johnson et al. 1988). Yannakakis (1997) shows that if Π is strongly NP-hard (NP-hard), then η cannot have the GS property unless P = NP (NP = co-NP). Under the assumption that P $\neq$ NP, since Maximum 3-Satisfiability is strongly NP-hard, the Maximum 3-Satisfiability result reported here implies that any neighborhood function (which can be searched in polynomial time) with the GS property for Maximum 3-Satisfiability must be data *dependent*. Also, the 0-1 Knapsack (TSP) result reported here implies that any neighborhood function (which can be computed in polynomial time) with the GS property for 0-1 Knapsack (TSP) must have size $\Theta(2^n)$ ($\Omega(2^{n/3})$) or else it must be data dependent, unless NP = co-NP (P = NP).

To describe the results, several discrete optimization problems are now formally described. For ease of notation, let $\boldsymbol{x}$ denote a non-negated literal and $\bar{\boldsymbol{x}}$ denote the corresponding negated literal. Therefore, a truth assignment t satisfies the literal $\boldsymbol{x}$ ($\bar{\boldsymbol{x}}$) if and only if $t(\boldsymbol{x})$ = T ($t(\boldsymbol{x})$ = F).

Maximum Satisfiability: Given m clauses, over n Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, find a truth assignment $t : X \rightarrow \{T,F\}$ that maximizes the number of satisfied clauses.

Maximum 3-Satisfiability is a special case of Maximum Satisfiability in which each clause has exactly three literals. Given a knapsack with a finite capacity and a (finite) collection of items, where each item has two integers associated with it (i.e., size and value), the objective of 0-1 Knapsack is to identify a subset of items that fit into the knapsack and have highest value. Instances of 0-1 Knapsack are formulated with respect to the definition of a consistent optimization problem.

0-1 Knapsack: Given vectors $\boldsymbol{s} = (s(1), s(2), \ldots, s(n))$, $\boldsymbol{v} = (v(1), v(2), \ldots, v(n))$, where $s(i), v(i) \in Z^+$, and capacity $B \in Z^+$, find the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ that maximizes the objective function $-\sum_{i=1}^{n} v(i) \max\{0, \sum_{i=1}^{n} x_i s(i) - B\} + \sum_{i=1}^{n} x_i v(i)$.

In this definition, $s(i)$ denotes the size of item i, $v(i)$ denotes the value of item i, and B denotes the size of the knapsack. The term $-\sum_{i=1}^{n} v(i) \max\{0, \sum_{i=1}^{n} x_i s(i) - B\}$ is a penalty function that ensures that any solution of 0-1 Knapsack, for which the collection of items exceeds the size of the knapsack, will have a nonpositive objective function value.

Symmetric TSP is now formally stated.

Traveling Salesman Problem (TSP): Given a collection of n cities $\{x_1, x_2, \ldots, x_n\}$ and distances $d(x_i, x_j)$ for each pair of cities x_i and x_j, where $x_i \neq x_j$, find a Hamiltonian circuit (permutation of the n cities, $y_1 y_2 \ldots y_n$, where for each i, $y_i = x_j$ for some j and $y_i \neq y_k$ for all $i \neq k$) with smallest total length $\left(d(y_1, y_n) + \sum_{i=1}^{n-1} d(y_i, y_{i+1}) \right)$.

Results on the size of data independent neighborhood functions for Maximum 3-Satisfiability, 0-1 Knapsack and TSP that have the smooth property are reported. Theorem 9 implies that the only data independent neighborhood functions for Maximum 3-Satisfiability with the smooth property are the complete neighborhood functions.

Theorem 9 (Armstrong and Jacobson 2005): If η is a data independent neighborhood function with the smooth property for Maximum 3-Satisfiability, then, for each instance I of Maximum 3-Satisfiability and truth assignment t over $n \geq 4$ variables, $\eta(I, t)$ consists of all truth assignments over the n variables, except for the truth assignment t itself.

In contrast to the result for Maximum 3-Satisfiability, there exists a data independent neighborhood function with the smooth property for 0-1 Knapsack that is not complete. The size of data independent neighborhood functions for 0-1 Knapsack can be given as a function of the number of possible items. Theorem 10 shows that there exists a data independent neighborhood function η^K with the smooth property for 0-1 Knapsack that has size $\Theta(2^n)$.

Theorem 10 (Armstrong and Jacobson 2005): There exists a data independent neighborhood function with the GS property for 0-1 Knapsack with size $f(n) = \begin{cases} 2^n - 2^{n/2+1} + n - k + 2, \text{for } n \text{ even} \\ 2^n - 2^{(n-1)/2} - 2^{(n+1)/2} + n - k + 2, \text{for } n \text{ odd} \end{cases}$, for $n \geq 1$, where n denotes the number of possible items.

Theorem 11 shows that the neighborhood function η^K described in the proof of Theorem 10 is the minimal neighborhood function with the smooth property. Therefore, Theorem 11 implies that a data independent neighborhood function with the smooth property for 0-1 Knapsack must have size $\Theta(2^n)$.

Theorem 11 (Armstrong and Jacobson 2005): Let η^K be the neighborhood function for 0-1 Knapsack that is given in the proof of Theorem 2. If η is a data independent neighborhood function such that $\eta(I, \boldsymbol{x}) \subset \eta^K(I, \boldsymbol{x})$

for some $x \in \{0,1\}^n$ ($n \geq 1$) and instance I, then η does not have the smooth property.

Theorem 12 shows that for every reasonable neighborhood function of TSP, there exists an instance of TSP with *strict* local optima that are not global optima; hence TSP has no reasonable neighborhood function with the smooth property. Furthermore, Theorem 12 shows that many exponentially-sized and data independent neighborhood functions do not have the smooth property. The proof of Theorem 12 follows by starting with an arbitrary solution ω and choosing another solution $\omega_{A'}$ (that is not a neighbor of ω) from an exponential set of solutions A' (Hamiltonian circuits) such that there does not exist any solution using edges from only ω and $\omega_{A'}$, except for the solutions ω and $\omega_{A'}$ themselves. The distances between the cities are then defined so that $\omega_{A'}$ and ω are the unique global optimum and unique second best solution (Hamiltonian circuit), respectively. For TSP, the size of data independent neighborhood functions can be given as a function of the number of cities n in an instance.

Theorem 12 (Armstrong and Jacobson 2005): If η is a data independent neighborhood function for the TSP with the smooth property, then $\min_{x \in SOL(I)} |\eta(I, x)| = \Omega(2^{n/3})$ where n denotes the number of cities.

Similar to Theorems 9 and 10, the class of instances used in the proof of Theorem 11 can be formulated into a polynomially solvable sub-problem. It follows that there exists a polynomially solvable sub-problem of TSP such that a data independent neighborhood function η with the smooth property must satisfy $\min_{x \in SOL(I)} |\eta(I, x)| = \Omega(2^{n/3})$. All these results demonstrate a drawback of local search algorithms that use data independent neighborhood functions for Maximum 3-Satisfiability, 0-1 Knapsack and TSP. These results also provide a first step towards showing that a large class of NP-hard discrete optimization problems has the property that every reasonable neighborhood function does not have the smooth property.

A difficulty with local search algorithms is that neighborhood functions for NP-hard discrete optimization problems typically have many (strict) local optima that are not global optima. The results reported here show that a large class of neighborhood functions for Maximum 3-Satisfiability, 0-1 Knapsack and TSP do not have the smooth property. In particular, the complete neighborhood functions are shown to be the only data independent neighborhood functions with the smooth property for Maximum 3-Satisfiability. The smallest data independent neighborhood function for 0-1 Knapsack is proven to have size with the same order of magnitude as the solution space size. Furthermore, the results demonstrate the minimal data independent neighborhood functions with the smooth property for 0-1 Knapsack and Maximum 3-Satisfiability. Every reasonable neighborhood function (and many exponentially sized data independent neighborhood functions) for TSP is shown to not have the smooth property.

Armstrong and Jacobson (2006a) examine the difficulty of finding neighborhood functions with a polynomial number of local optima and where the length of improving paths are bounded above by a polynomial in the size of the problem instance. They also considers neighborhood functions for which the order of local optima are bounded above by a polynomial in the size of the problem instance. For an instance I of a discrete maximization problem with neighborhood function η and objective function m, an improving path of length n is a sequence of solutions $s_0, s_1, \ldots, s_n$ such that $s_i \in \eta(s_{i+1})$ and $m(s_i) < m(s_{i+1})$ for all $i = 0,1,\ldots,n-1$. The *order* of a solution s is the rank of the solution in terms of the objective function values. For example, if I is an instance of maximization problem A with solution space SOL(I) and objective function m such that $m(\text{SOL}(I)) = \{4,7,9,12\}$, then any solution s such that $m(s) = 12$ has order one, any solution s with $m(s) = 9$ has order two, any solution s with $m(s) = 7$ has order three, and any solution s with $m(s) = 4$ has order four. A neighborhood function that can be searched in polynomial time (in the size of the problem instance) for an improving solution, or else correctly concludes that no such solution exists, is called a *polynomially computable neighborhood function*. A neighborhood function whose size is polynomial in the length of the problem instance and independent of the problem data is termed *reasonable* [10] (see below for a formal definition). Armstrong and Jacobson (2006a) show that neighborhood transformations and data-independent order transformations (DIOTs) preserve the length of improving paths and the order of local optima. For example, if problem A (DIOT) neighborhood transforms to problem B, then for any (reasonable) polynomially computable neighborhood function η_B for B, there exists a (reasonable) polynomially computable neighborhood function η_A for A such that for each instance I of A, an instance $f(I)$ of B can be constructed in polynomial-time, where for every improving path of I with respect to neighborhood function η_A, there exists a corresponding improving path of $f(I)$ with respect to the neighborhood function η_B of the same length. Furthermore, given an improving path (with respect to neighborhood η_A) for I, the solutions on the corresponding improving path (with respect to neighborhood η_B) for $f(I)$ will be of the same order. This implies that if there exists a improving path (with respect to neighborhood η_A) for instance I of A that terminates at a local optimum with order p, then there exists a corresponding improving path (with respect to neighborhood η_B) for instance $f(I)$ of problem B that also terminates at a solution with order p.

The objective of this research is to analyze the difficulty in addressing hard discrete optimization problems with local search algorithms. In particular, its focus is to study the complexity of finding neighborhood functions with desirable properties for NP-hard discrete optimization problems. One desirable property for a neighborhood function is to have few local optima that are not global optima, thereby smoothing out the landscape of the solution space and increasing the likelihood that an improving solution can be found in the neighborhood of a given solution. The term (strict) L-local is used to represent a local optimum that is not a global optimum. Ideally a neighborhood function would have few L-locals, each of which with a relatively small order. The notion of few L-locals and small order are (for the purposes here) encapsulated by the following definitions. Given a discrete optimization problem A and an associated instance I, a neighborhood function η for A is said to be (strict) *local bounded* if the number of (strict) L-locals is bounded above by a polynomial in the size of I. A (strict) order bounded neighborhood function has (strict) local optima with order bounded above by a polynomial in the size of I. Another desirable property for a neighborhood function is the notion of finding local optima quickly or in polynomial time, which is addressed by analyzing the complexity of neighborhood functions with improving paths of polynomial length. A neighborhood function η for problem A is said to be *compact* if for all instances I and all solutions s (of I), every improving path from s to a L-local has length that is bounded above by a polynomial in the size of I. Also, a polynomially computable neighborhood function η for problem A is said to be *semi-compact* if for all instances I and all solutions s (of I), there exists an improving path from s to an L-local that has length bounded by a polynomial in the size of I.

Armstrong and Jacobson (2006b) defined order transformations such that the ordering imposed by the objective function is preserved through the transformation. They also introduced neighborhood transformations between optimization problems that preserve the number of L-locals for polynomially computable neighborhood functions. These results focus on the *number* of L-locals rather than the specific local optima, since the difficulty in addressing hard discrete optimization problems with local search algorithms often arises out of the number of L-locals and not the number of local optima (since they include all global optima). Armstrong and Jacobson (2006b) show that if problem A neighborhood transforms to problem B and B has a local bounded and polynomially computable neighborhood function, then A has a local bounded and polynomially computable neighborhood function. MAX Clause Weighted SAT (MCWS) and Zero-One Integer Programming (ZOIP) are proven to be NPO-complete with respect to neighborhood transformations in Armstrong and Jacobson (2006b). MCWS is a generalization of Maximum Satisfiability, where there is a weight w_i ($i = 1,2,\ldots,m$) associated with each clause and the goal is to find a truth assignment that maximizes the sum of the weights of the satisfied clauses. Armstrong and Jacobson (2004) introduced another type of order transformation, called a data-independent order transformation (DIOT), which preserves the number of L-locals for reasonable neighborhood functions.

The results reported here extend those reported in Armstrong and Jacobson (2004, 2006b) by investigating how the transformations preserve the length of improving paths. Here, neighborhood transformations and DIOTs are investigated to show how they preserve improving paths of particular neighborhood functions. In particular, we show that if there exists a (semi-) compact neighborhood function for either MCWS or ZOIP, then every problem in NPO has a (semi-) compact neighborhood function. Also, if there exists a (strict) order bounded neighborhood function for either MCWS or ZOIP, then every problem in NPO has a (strict) order bounded neighborhood function. An example of a neighborhood transformation from Traveling Salesman Problem (TSP) to MCWS is also presented, together with an explanation of how the transformation can be used to define a neighborhood function (or local search algorithm) for TSP given a neighborhood function for MCWS.

There are a number of other insightful articles related to the material presented here. Johnson et al. (1988) introduce the class of problems PLS and investigate the complexity of finding local optima for neighborhood functions of discrete optimization problems. The class of problems PLS contains all local search problems (discrete optimization problems together with a neighborhood function) in which local optimality can be verified in polynomial time. The results reported here focus on the *existence* of a neighborhood function with a polynomial bound on the length of improving paths, while Johnson et al. (1988) focus on finding local optima in polynomial time for a *given* neighborhood function. If a discrete optimization problem has a polynomially computable and compact neighborhood function, then the corresponding local search problem must be in PLS. Note that although a given neighborhood function η is semi-compact and polynomially computable, it is not guaranteed that local optima of η can be found in polynomial time, since for a given solution s there can be exponentially many improving paths to L-locals such that only a few of these improving paths have length that is polynomial in the size of the problem instance. However, it is interesting to study semi-compact neighborhood functions since if a neighborhood function is not semi-compact, then local search *will not* find local minima in polynomial time. Therefore, semi-compactness of a neighborhood function is a desirable property.

Jacobson and Solow (1993) and Armstrong and Jacobson (2005) investigate the complexity of finding polynomial time improvement algorithms for discrete optimization problems. A *polynomial time improvement algorithm* is a polynomial-time algorithm such that given any solution, either another solution can be found with better objective function value or else the algorithm concludes that no such solution exists and the given solution is a global optimum. A polynomial time improvement algorithm is equivalent to a polynomially computable neighborhood function with zero L-locals. Armstrong and Jacobson (2005) investigate the existence of reasonable neighborhood functions, for NP-hard discrete optimization problems, in which there exists an improving path from every solution to a global optimum. The work presented here does not examine the existence of improving paths to global optima for neighborhood functions; rather, it considers upper bounds on the length of improving paths to local optima.

Research on approximation preserving reductions (Ausiello et al. 1995) has some similarities with the work presented here. The set of problems APX consist of all problems A in NPO in which, for some $\varepsilon > 0$, there exists a polynomial time ε-approximate algorithm (Papadimitriou and Steiglitz 1982). The set of problems PTAS (polynomial time approximation scheme) contains all problems in NPO in which there exists a polynomial time ε-approximate algorithm, for all $\varepsilon > 0$. Several reductions between discrete optimization problems have been defined to preserve inclusion in APX or PTAS. For example, Crescenzi and Trevisan (2000) introduce a new approximation preserving reduction, called PTAS-reducibility, which generalizes other transformations that preserve approximations. They show that if A PTAS-reduces to B and $B \in$ PTAS, then $A \in$ PTAS. Crescenzi and Trevisan (2000) also show that Maximum Satisfiability is APX-complete (Maximum Satisfiability $\in$ APX and for all $A \in$ APX, A PTAS-reduces to Maximum Satisfiability) under the PTAS-reducibility. Some problems have been shown to be NPO-complete with respect to an approximation preserving reduction. Ausiello et al. (1995) show that MAX Weighted Boolean SAT is NPO-complete with respect to an approximation preserving reduction. The main difference between these articles and the results presented here are the type of reductions being considered (order preserving reductions rather than approximation preserving reduction).

Several definitions are needed to describe the results. The set of NPO problems are the set of all discrete optimization problems A in which the corresponding decision problem of A is in NP (Auisello et al. 1995). An NPO problem A is a four-tuple (D, SOL, m, goal) where D is the set of instances of A, SOL(I) is the set of solutions for instance $I \in D$, $m(I, s)$ is the objective function for solution $s \in$ SOL(I), and goal is equal to "min" or "max" depending on the type of optimization problem. For a problem to be in NPO, several conditions must be satisfied. The set of instances D and solutions $s \in$ SOL(I) must be recognizable in polynomial time in the size of the problem instance I. Furthermore, the objective function needs to be computable in polynomial time in the size of I. Throughout the remainder of this discussion, assume that all discrete optimization problems are maximization problems. Therefore, any problem in NPO can be represented by the three-tuple (D, SOL, m).

For every instance I of problem $A = (D, \text{SOL}, m)$, a *neighborhood function* $\eta(I, s) \subseteq$ SOL(I) maps each solution $s \in$ SOL(I) into a subset of the solution space. The *size* of a neighborhood function η on instance I is $\max_{s \in SOL(I)} |\eta(I, s)|$. For an instance I of a maximization problem, a solution $s \in$ SOL(I) is a (*strict*) *local optimum* if $m(I, s)$ (>) $\geq m(I, s')$ for all $s' \in \eta(I, s)$, and a solution $s \in$ SOL(I) is a *global optimum* if $m(I, s) \geq m(I, s')$ for all $s' \in$ SOL(I). A solution s is a (strict) L-local if s is a (strict) local optimum that is not a global optimum.

For every discrete optimization problem $A = (D, \text{SOL}, m)$ in NPO, there exists an increasing sequence of positive integers $\{n_i\}_{i=1}^{\infty}$ and a corresponding sequence of sets $\{S_i\}_{i=1}^{\infty}$ such that for each $i \in Z^+$, $S_i \subseteq \{0,1\}^{n_i}$ defines a solution space for an instance of A. For a given discrete optimization problem, the sequences $\{n_i\}_{i=1}^{\infty}$ and $\{S_i\}_{i=1}^{\infty}$ are defined according to some encoding scheme (Garey and Johnson 1979) used for the solution space. Therefore, for a given discrete optimization problem, the sequences $\{n_i\}_{i=1}^{\infty}$ and $\{S_i\}_{i=1}^{\infty}$ may be defined differently. However, any two different pairs of sequences $\{n_{1i}\}_{i=1}^{\infty}$ and $\{n_{2i}\}_{i=1}^{\infty}$ are defined such that the rate of growth of one of the sequences is polynomial in the rate of growth of the other sequence. In particular, there exists two polynomials p_1 and p_2 such that $n_{1i} \leq p_1(n_{2i})$ and $n_{2i} \leq p_2(n_{1i})$ for all $i \in Z^+$. For every $i \in Z^+$, there exists a set of instances D_i such that $I \in D_i$ if and only if SOL(I) = S_i. Therefore, A can be represented as the four-tuple $(\{D_i\}_{i=1}^{\infty}, \{S_i\}_{i=1}^{\infty}, \{n_i\}_{i=1}^{\infty}, m)$. This notation modification is used throughout the rest of the discussion.

A *reasonable neighborhood function* (Tovey 1985) for a discrete optimization problem $A = (\{D_i\}_{i=1}^{\infty}, \{S_i\}_{i=1}^{\infty}, \{n_i\}_{i=1}^{\infty}, m)$ is defined to be a neighborhood function that is polynomial in n_i and independent of the problem data. That is, a reasonable neighborhood function η satisfies:

(1) $|\eta(I, s)| \le p(n_i)$ for all $i \in Z^+$, $I \in D_i$, and $s \in S_i$, where p is a polynomial function.

(2) For each $i \in Z^+$, $\eta(I_1, s) = \eta(I_2, s)$ for all $s \in S_i$ and $I_1, I_2 \in D_i$,

Let A and B be two maximization problems. The problem $A = (D_A, \mathrm{SOL}_A, m_A)$ *order transforms* to $B = (D_B, \mathrm{SOL}_B, m_B)$, if there exists two functions f and q that satisfy the following:

(1) $f(I) \in D_B$ for all $I \in D_A$ and f is computable in polynomial time in the size of I.

(2) For each $I \in D_A$ and $s \in \mathrm{SOL}_A(I)$, $q(I, s) \in \mathrm{SOL}_B(f(I))$. Also, if $s' \in \mathrm{SOL}_A(I)$ and $m_A(I, s)$ (>) $\ge m_A(I, s')$, then $m_B(f(I), q(I, s))$ (>) $\ge m_B(f(I), q(I, s'))$,

(3) For each $I \in D_A$, $q(I,.)$ is a one-to-one function and it can be determined (in polynomial time in the size of I) if a solution $s_B \in \mathrm{SOL}_B(f(I))$ is also in $q(I, \mathrm{SOL}_A(I))$,

(4) Let $\Omega = q(I, \mathrm{SOL}_A(I))$ be the set of solutions to problem B, instance $f(I)$, that are "mapped" from instance I of problem A. Also, let $\Omega^c = \mathrm{SOL}_B(f(I)) - q(I, \mathrm{SOL}_A(I))$ be the set of solutions to instance $f(I)$ of problem B that are not "mapped" from solutions of instance I. Then $\min_{s \in \Omega}[m_B(f(I), s)] \ge \max_{s \in \Omega^c}[m_B(f(I), s)]$.

Condition (2) guarantees that for each instance I of A, the instance $f(I)$ preserves the ordering of the solutions in I. Condition (3) guarantees that the transformed solutions can be recognized in polynomial time in the size of instances of problem A. Condition (4) guarantees that for each instance I, the set of solutions that are not in $q(I, \mathrm{SOL}_A(I))$ have an objective function value no larger than any solution in $q(I, \mathrm{SOL}_A(I))$. This condition forces the local structure of instance $f(I)$ to be the same as the local structure of instance I, except that instance $f(I)$ may have more solutions than instance I, as long as these extra solutions (that are in $\mathrm{SOL}_B(f(I)) - q(I, \mathrm{SOL}_A(I))$) have objective function values that are less than any solution in $q(I, \mathrm{SOL}_A(I))$. If the function q and its inverse are computable in polynomial time in the size of I, the transformation is said to be a *neighborhood transformation* (discussed in detail in Armstrong and Jacobson (2006b). An example of a neighborhood transformation from TSP to MCWS is given below. The problem $A \in$ NPO is said to be *NPO-complete with respect to a transformation* (denoted by $\propto$) if for all problems $B \in$ NPO, $B \propto A$.

Suppose $A = (\{D_{Ai}\}_{i=1}^{\infty}, \{S_{Ai}\}_{i=1}^{\infty}, \{n_{Ai}\}_{i=1}^{\infty}, m_A)$ *order transforms* to $B = (\{D_{Bi}\}_{i=1}^{\infty}, \{S_{Bi}\}_{i=1}^{\infty}, \{n_{Bi}\}_{i=1}^{\infty}, m_B)$ where f (q) denote the transformation of instances (solutions) of A to instances (solutions) of B. By the definition of order transformation, for each instance $I \in D_{Ai}$ there exists an instance $f(I) \in D_{Bj}$ for some j. The order transformation is called data-independent if the function q is independent of the problem data of A and the function f satisfies: for every $i \in Z^+$, there exists $j \in Z^+$ such that for all $I \in D_{Ai}$, $f(I) \in D_{Bj}$. Note that all the transformations defined here are transitive.

Armstrong and Jacobson (2006b) show that if A neighborhood transforms to B and B has a local bounded and polynomially computable neighborhood function, then so does problem A. They also show that if problem A DIOT to problem B and B has a local bounded and reasonable neighborhood function, then so does problem A. These results are extended to show that neighborhood transformations and DIOTs preserve the length of *every* improving path. Suppose that A neighborhood or data-independent order transforms to B with transformation functions f and q. Given a neighborhood function η for B, a neighborhood function is defined for A as follows: For each instance $I_A \in D_A$ and $s \in \mathrm{SOL}(I_A)$, let $\eta_{f,q}(I_A, s) = \{s' : q(I_A, s') \in \eta(f(I_A), q(I_A, s))\}$. By definition, if f and q represent a neighborhood transformation, then neighborhood $\eta_{f,q}$ is polynomially computable whenever η is polynomially computable (Armstrong and Jacobson 2006b). Similarly, if f and q represent a DIOT and η is a reasonable neighborhood function for B, then $\eta_{f,q}$ is a reasonable neighborhood function for A (Armstrong and Jacobson 2004).

Theorem 13 (Armstrong and Jacobson 2006a)**:** Let A and B be two discrete optimization problems such that A neighborhood transforms to B. If B has a local bounded, semi-compact, and polynomially computable neighborhood function, then A has a local bounded, semi-compact, and polynomially computable neighborhood function.

The results reported here are for neighborhood functions that are locally bounded. The "local bounded" term could be removed from all the theorems given below and the augmented result would still hold, since the transformations preserve local bounded neighborhood functions, order bounded neighborhood functions, and (semi) compact neighborhood functions independently. The results are given for local bounded neighborhood functions since the neighborhood transformation and DIOT do not preserve connected neighborhood functions (i.e., there exists a path between all pairs of solutions where each solution in the path is a neighbor of the preceding solution). In other words, the neighborhood function $\eta_{f,q}$ may be disconnected when η is a connected neighborhood function. Note that it is straightforward to construct disconnected neighborhood functions for an optimization problem with a polynomial bound on the length of improving paths (for example, consider the neighborhood function $\eta_{\varnothing}$, where $\eta_{\varnothing}(I,s) = \varnothing$ for all instances I and solutions s). Therefore, the results would not be interesting with the "local bounded" term removed. The proof of

Theorem 14 follows from Theorem 13, with the exception that the neighborhood function $\eta_{f,q}$ is reasonable if η is reasonable (see Armstrong and Jacobson (2004).

Theorem 14 (Armstrong and Jacobson 2006a): Let A and B be two discrete optimization problems in NPO such that A DIOT to B. If B has a local bounded, semi-compact, and reasonable neighborhood function, then problem A has a local bounded, semi-compact, and reasonable neighborhood function.

Theorems 15 and 16, which extend Theorems 13 and 14, are concerned with an upper bound on the length of every improving path.

Theorem 15 (Armstrong and Jacobson 2006a).: Let A and B be two discrete optimization problems such that A neighborhood transforms to B. If B has a local bounded, compact, and polynomially computable neighborhood function, then problem A has a local bounded, compact, and polynomially computable neighborhood function.

The proof of Theorem 16 follows exactly from Theorem 15, after noting that the function $\eta_{f,q}$ is reasonable if η is reasonable (Armstrong and Jacobson (2004). Since the function q of a neighborhood transformation (from a problem A to a problem B) preserves the order of the solutions between the problems, the order (or rank) of the final solutions along improving paths are preserved between corresponding improving paths. In other words, suppose problem A neighborhood transforms to B and η is a neighborhood function for problem B. Given an improving path (with respect to neighborhood function $\eta_{f,q}$) between solution s and local optimum s' of instance I_A of problem A, there exists an improving path (with respect to neighborhood function η) between $q(I_A,s)$ and $q(I_A,s')$ of problem B such that the local optimum $q(I_A,s')$ has the same order as the local optimum s' for problem A.

Theorem 16 9 Armstrong and Jacobson 2006a): Let A and B be two discrete optimization problems such that A DIOT to B. If B has a local bounded, compact, and reasonable neighborhood function, then problem A has a local bounded, compact, and reasonable neighborhood function.

Theorem 17 shows that neighborhood transformations preserve (strict) order bounded neighborhood functions; it is a restatement of Theorem 13 with the term "(strict) order bounded" used in place of the term "semi-compact".

Theorem 17 (Armstrong and Jacobson 2006a): Let A and B be two discrete optimization problems in NPO such that A neighborhood transforms to B. If B has a local bounded, (strict) order bounded, and polynomially computable neighborhood function, then A has a local bounded, (strict) order bounded, and polynomially computable neighborhood function.

Note that Theorem 17 may be restated with the terms "neighborhood transforms" and "polynomially computable" replaced by the terms "DIOT" and "reasonable." The proof and statement of this theorem are omitted, where the proof follows analogously to the proof of Theorem 17. Also, Theorem 17 holds when the "local bounded" term is removed. Each of the following theorems demonstrate how a different neighborhood property is preserved by the neighborhood transformation or DIOT. The results reported here and in Armstrong and Jacobson (2006b), show that if A (DIOT) neighborhood transforms to B and B has a neighborhood with property p, then A has a neighborhood with property p; where p can be any of the following: local bounded, strict local bounded, semi-compact, compact, order bounded, strict order bounded, (reasonable) polynomially computable. The results imply that a neighborhood function with these good properties for one problem may be transformed (via a neighborhood transformation or DIOT) to a neighborhood function for another problem with the same properties.

To demonstrate how the neighborhood transformation or DIOT can be used to define an effective neighborhood function for one problem given a good neighborhood function for another problem, a neighborhood transformation is given from TSP to MCWS.

Example: Neighborhood Transformation from TSP to MCWS.

By labeling the cities of a n-city TSP instance as $1,2,\ldots,n$, then any instance can be represented by the set $\{d_{ij} : i,j = 1,2,\ldots,n; j > i\}$ of distances between each pair of cities (d_{ij} equals the distance between city i and city j). To specify a neighborhood transformation to MCWS, a function f is given that maps instances of TSP to instances of MCWS, and another function q is given that maps solutions of TSP to solutions of MCWS. Given a n-city TSP instance I, define $f(I)$ to be the set of Boolean variables $X=\{x_{ij} : i,j = 1,2,\ldots,n; j > i\}$, with the following clauses and weights (to simplify notation let x_{ji} represent x_{ij} when $j > i$):

$$\text{clause } (x_{ij}) \text{ with weight } -d_{ij} \text{ for all } i,j = 1,2,\ldots,n; j > i\ ;$$

$$\text{clause } (x_{i1}, x_{i2},\ldots, x_{i,i-1}, x_{i,i+1},\ldots, x_{in}) \text{ with weight } M = (n+1)\max\{d_{ij}\} \text{ for all } i = 1,2,\ldots,n\ ;$$

$$\text{clause } (x_{i1}, x_{i2},\ldots, x_{i,i-1}, x_{i,i+1},\ldots, \bar{x}_{ij},\ldots, x_{in}) \text{ with weight } M \text{ for all } i,j = 1,2,\ldots,n; j \neq i\ ;$$

$$\text{clause } (\bar{x}_{ij}, \bar{x}_{ik}, \bar{x}_{il}) \text{ with weight } M \text{ for } i,j,k,l = 1,2,\ldots,n; j \neq i, k \neq i, l \neq i, k \neq j, l \neq j, l \neq k.$$

The transformation of solutions q maps a tour φ of instance I to a truth assignment $t : X \rightarrow \{0,1\}^{|X|}$, where $t(x_{ij}) = 1$ if and only if city j follows city i or city i follows city j in tour φ. Let η represent the 4-flip

neighborhood function of MCWS, then $\eta_{f,g}$ is the 2-change neighborhood function of TSP. In general, the (4+2x)-flip neighborhood of MCWS, where x = 0,1,2,..., corresponds, via the given neighborhood transformation, to the (2+x)-change neighborhood of TSP. Now, given an effective procedure for searching the solution space of MCWS with the (4+2x)-flip neighborhood function, for some x = 0, 1, 2,..., an effective procedure is obtained for searching the (2+x)-change neighborhood function of TSP. In this sense, the term effective represents procedures that find local optima in a few number of iterations and/or finds solutions with low order.

5. Threshold Analysis Performance

The threshold analysis framework for analyzing the performance of generalized hill climbing algorithms has been introduced. Initial results with this framework are reported in Jacobson et al. (2005a).

For hard discrete optimization problems, local search algorithms have been formulated with the hope of finding good or near-optimal solutions. Generalized hill climbing (GHC) algorithms (Jacobson et al. 1998, Johnson and Jacobson 2002a,b) provide a framework for local search algorithms that can be applied to a wide variety of hard discrete optimization problems. Many well-known local search algorithms can be modeled within the GHC framework, including simulated annealing (SA) (Kirkpatrick et al. 1983) and threshold accepting (Dueck and Scheuer 1990). The objective of all these algorithms is to find the best possible solution using a limited amount of computing resources. A further challenge is to construct algorithms that find near-optimal solutions for all instances of a particular problem, since the effectiveness of many algorithms tends to be problem-specific, as they exploit particular characteristics of problem instances (e.g., Lin and Kergnihan 1973 for the travelling salesman problem). It is therefore important to assess the performance of algorithms and devise strategies to improve their effectiveness in solving hard discrete optimization problems.

There are several results in the literature concerning the asymptotic performance of SA. For SA with an exponential acceptance probability function, Mitra et al. (1986) and Hajek (1988) develop conditions for three convergence properties: asymptotic independence of the starting conditions, convergence in distribution of the solutions generated, and convergence to a global optimum; they also characterize the convergence rate. Anily and Federgruen (1987) extend these results to SA with general acceptance probabilities by developing necessary and sufficient conditions for convergence, and provide conditions for the reachability of the set of global optima. Yao and Li (1991) and Yao (1995) also discuss SA with general acceptance probabilities, though their primary contribution is with respect to general neighborhood generation distributions. Ferreira and Zerovnik (1993) develop bounds on the probability that SA obtains an optimal / near-optimal solution. They also show that random restart local search dominates SA, measured by the probability of finding a globally optimal solution, as the number of restarts grows. Similar results are reported in Jacobson and Yucesan (2004a,b). For a comprehensive survey of SA convergence results, see Aarts and Korst (2002) or Henderson et al. (2003).

There have been a limited number of results on finite-time performance measures for SA. Mitra et al. (1986) present bounds for the objective function over a finite horizon. Chiang and Chow (1989) and Mazza (1992) investigate the statistical properties of the first visit time to a global optimum that provides insight into the time-asymptotic properties of the algorithm as the outer loop counter approaches infinity. Cantoni and Cerf (1997) investigate optimizing a finite-horizon cooling schedule to maximize the number of visits to a global optimum after a finite number of iterations. Desai (1999) focuses on finite-time performance by incorporating size-asymptotic information supplied by certain eigenvalues associated with the transition matrix. Desai provides quantitative and qualitative information about the performance of SA after a finite number of steps by observing the quality of solutions related to the number of steps that the algorithm has taken.

Srichander (1995) examines the finite-time performance of SA using spectral decomposition of matrices. Srichander shows that for the final solution of SA to converge to the global minimum with probability one, an annealing schedule on the temperature is not necessary. Furthermore, Srichander shows that annealing schedules on the temperature produce an inefficient algorithm in that the number of function evaluations required to obtain a global minimum is extremely large. Srichander also presents a modified SA algorithm with an iterative schedule on the size of the neighborhood sets that leads to a more efficient algorithm. The performance of this algorithm on a real-world example is reported. Fleischer and Jacobson (1999) present an entropy measure of the Markov chain model for SA, and use this measure to compare several finite-time SA implementations, with different cooling schedules. They show that higher values for this entropy measure correspond to better finite-time performance, as measured by the likelihood of terminating in a globally optimal solution. Nolte and Schrader (2001) use results on rapidly mixing Markov chains to obtain bounds for the stationary distribution of the globally optimal solutions.

Orosz and Jacobson (2002a) report finite-time performance measures for GHC algorithms by analyzing a GHC algorithm's ability to visit solutions that are *close enough* to a global minimum, as measured by the

objective function value. Solutions that have objective function value less than or equal to β are referred to as *β-acceptable solutions*, where β denotes the objective function value (typically larger than the global minimum objective function value) that should be visited. This report summarizes research presented in Jacobson et al. (2005a). They provide a closed form expression for the expected number of iterations to visit a β-acceptable solutions for the first time for cyclical simulated annealing (CSA), an algorithm that executes in the same manner as simulated annealing, except that the cooling schedule cycles through a prespecified set of temperature values. They also show how the expected number of iterations to visit β-acceptable solutions can be estimated for such algorithms.

Complete details of the GHC algorithm framework can be found in Jacobson et al. (1998). To describes the resulted presented here, several definitions are needed. For a discrete (minimization) optimization problem, define the *solution space*, Ω, as a finite set of all possible solutions. Define an *objective function* f: $\Omega \rightarrow [0, +\infty)$ that assigns a non-negative value to each element of the solution space. Two important components of GHC algorithms are the *neighborhood function*, η: $\Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subseteq \Omega$ for all $\omega \in \Omega$, and the *hill climbing random variables* R_k: $\Omega \times \Omega \rightarrow \Re$, $k = 1,2,\ldots$. For each solution $\omega \in \Omega$, the neighborhood function $\eta(\omega)$ defines a set of solutions that are close to ω (Aarts and Korst 2002). The neighborhood function is assumed to be symmetric (i.e., if $\omega' \in \eta(\omega'')$, then $\omega'' \in \eta(\omega')$ for all $\omega', \omega'' \in \Omega$) and that $\omega \in \eta(\omega)$ for all $\omega \in \Omega$. Moreover, at each iteration of a GHC algorithm, a solution is randomly generated among all neighbors of the current solution by a neighborhood probability mass function, where the resulting random variables are conditionally independent (on the current solution). For example, neighbors are said to be generated uniformly at each iteration of a GHC algorithm execution if, for all $\omega \in \Omega$, with $\omega' \in \eta(\omega)$,

$$P\{\omega' \text{ is selected as the neighbor of } \omega \text{ at a given iteration of a GHC algorithm}\} = 1 / |\eta(\omega)|.$$

Unless otherwise stated, assume that neighbors are generated uniformly at each GHC algorithm iteration.
The hill climbing random variables, which are assumed to be independent, determine whether a randomly generated neighboring solution is accepted during a particular inner loop iteration associated with outer loop iteration k. Stopping criteria for the inner and outer loops (*STOP INNER* and *STOP OUTER*, respectively) determine when the hill climbing random variable index k increments by one, hence a new hill climbing random variable is used to accept or reject neighboring solutions. Although the range of the hill climbing random variables can be the set of real numbers, $\Re$, in practice they are typically restricted to the set of non-negative real numbers, $\Re^+$ (which is assumed for the rest of the discussion). Therefore, for minimization problems, when a randomly generated neighboring solution has objective function value greater than the current solution, then the neighboring solution is accepted (hence becomes the new current solution) if the difference between the objective function values is not too large (i.e., smaller than the value generated for the hill climbing random variable). This concept of accepting an inferior solution is the origin for the name "hill climbing".

The neighborhood function establishes relationships between the solutions in the solution space, hence allows the solution space to be traversed or searched by moving between solutions. To ensure that the solution space is not fragmented, we assume that all the solutions in the solution space (with neighborhood function η) are *reachable* (i.e., for all $\omega', \omega'' \in \Omega$, there exists a set of solutions $\omega_1, \omega_2, \ldots, \omega_m \in \Omega$ such that $\omega_j \in \eta(\omega_{j-1}), j = 1,2,\ldots,m+1$, where $\omega' \equiv \omega_0$ and $\omega'' \equiv \omega_{m+1}$). If all solutions in the solution space are reachable, then the solution space (with neighborhood function η) is said to be reachable.

The solution space for a discrete optimization problem can be partitioned into two mutually exclusive and collectively exhaustive sets:

- the set of globally optimal solutions, $G = \{\omega^* \in \Omega: f(\omega^*) \le f(\omega) \text{ for all } \omega \in \Omega\}$
- the set of all other solutions, $G^c = \{\omega \in \Omega: f(\omega^*) < f(\omega), \omega^* \in G\}$ where $G \cup G^c = \Omega$

A common goal for discrete optimization problem is to identify a globally optimal solution $\omega^* \in G$. However, from a practical point of view, solutions that are *close enough* to a globally optimal solution (where close enough is measured in terms of the objective function value) for a discrete optimization problem may be acceptable. To describe such solutions, define the set of *β-acceptable solutions*,

$$D_\beta = \{\omega \in \Omega: f(\omega) \le \beta\} \text{ for } \beta \in \Re. \tag{20}$$

Note that if $\beta < f(\omega^*)$, $\omega^* \in$ G, then $D_\beta = \varnothing$. Moreover, if $\beta \ge max_{\omega \in \Omega} f(\omega)$, then $D_\beta = \Omega$. Lastly, $\lim_{\beta \to f(\omega^*)^+} D_\beta = G$, hence G is the upper (right) limit of D_β as β approaches $f(\omega^*)$ from above.

Each execution of a GHC algorithm generates a sequence (sample) of solutions. In practice, the best solution obtained over the entire GHC algorithm run, not just the final solution, is reported. This allows the algorithm to aggressively traverse the solution space visiting many inferior solutions *en route* to a globally optimal solution, while retaining the best solution obtained throughout the entire GHC run. Without loss of

generality, assume that GHC algorithm runs are initialized at solutions not in D_β (i.e., $\omega(0) \in (D_\beta)^c$). Therefore, each sequence of solutions is a function of the initial solution and two independent sets of independently and identically distributed (IID) $U(0,1)$ random variables (i.e., uniformly distributed on the interval (0,1)),

i) $\{\xi_i\}$, that generate the neighbors, $\omega \in \eta(\omega(i))$, from the neighborhood probability mass function $g(\omega(i),.)$ (hence allows $\delta(\omega(i), \omega) = f(\omega) - f(\omega(i))$ to be computed) at each iteration $i = 1,2,\ldots$,

ii) $\{\nu_i\}$, that generate values for $R_k(\omega(i), \omega)$ (i.e., R) to determine whether the $\omega \in \eta(\omega(i))$ is accepted or rejected (i.e., $R \geq \delta(\omega(i), \omega)$ or $R < \delta(\omega(i), \omega)$, respectively).

In addition, assume that when comparing different GHC algorithms, the initial solution is given and fixed across all such algorithms. This means that the initial solution $\omega(0)$ and $(\{\xi_i\},\{\nu_i\})$ completely define the probability of each sequence of solutions generated by GHC algorithms. For simplicity and ease of notation, $\omega(0)$ and $(\{\xi_i\},\{\nu_i\})$ are suppressed, unless they are needed to avoid ambiguities.

Simulated annealing (SA) is a particular GHC algorithm, with hill climbing random variable, $R_k(\omega, \omega') = R_k = -T(k)*ln(1-U_k)$, $\omega' \in \eta(\omega)$, where the $\{U_k\}$ are IID $U(0,1)$ random variables and $T(k)$, $k = 1,2,\ldots$, are the temperate parameters that define the cooling schedule. Therefore, if $f(\omega) - f(\omega(i)) > 0$, then $\omega(i)$ is accepted as the current solution with probability $e^{-[f(\omega) - f(\omega(i))]/T(k)}$. The temperature parameters are usually set such that they gradually decrease to zero as k approaches infinity, where SA algorithm then behaves similar to pure local search algorithm. Hence, given enough iterations, it terminates at either a global $\omega^* \in G$, or a local optimum $\omega^* \in L = \{\omega \in \Omega: f(\omega') \geq f(\omega)$ for all $\omega' \in \eta(\omega)\}$.

Cyclical simulated annealing (CSA) is a particular type of SA algorithm, where the cooling schedule cycles through a predetermined set of temperatures during the algorithm's execution. To simplify the description and implementation of CSA, suppose that each middle loop is of fixed length h, which also corresponds to the length of each CSA temperature cycle, given by $T(1), T(2), \ldots, T(h)$, where $T(n+kh) = T(n+(k+1)h)$, $n = 1,2,\ldots,h$, $k = 0,1,\ldots$. The outer loop iterations correspond to the number of cycles executed, while the inner loop iterations correspond to the number of iterations executed at each of the h temperature values. The CSA algorithm is described in pseudo-code form:

CSA Algorithm
Inputs:
  Set the iteration index $i = 0$, $k = 1$
  Generate an initial solution $\omega(0) \in \Omega$ and set $\omega^* \leftarrow \omega(0)$
  Set the cycle length h and set a single cycle cooling schedule $T(1), T(2),\ldots, T(h)$
Repeat
  Do for $j = 1,2,\ldots,h$
  Repeat
  Generate a neighboring solution $\omega \in \eta(\omega(i))$ and a $U(0,1)$ random variate U
    Compute $\delta(\omega(i), \omega) = f(\omega) - f(\omega(i))$
    If $-T(j)\, ln(1-U) \geq \delta(\omega(i), \omega)$, then $\omega(i+1) \leftarrow \omega$
    If $-T(j)\, ln(1-U) < \delta(\omega(i), \omega)$, then $\omega(i+1) \leftarrow \omega(i)$
    If $f(\omega(i+1)) < f(\omega^*)$, *set* $\omega^* \leftarrow \omega(i+1)$
    $i \leftarrow i + 1$
  Until STOP INNER
  End Do
  $k \leftarrow k + 1$
  Until *STOP OUTER*
Output: Report ω^*

Orosz and Jacobson (2002a) introduce and analyze the *one-step β-acceptable probability* as a finite time performance measure for GHC algorithms. To describe this measure, consider a GHC algorithm applied to an instance of a discrete optimization problem, where $R_k(\omega(i),\omega) \geq 0$, $\omega(i) \in \Omega$, $\omega \in \eta(\omega(i))$, for all iterations $i = 1,2,\ldots$, and for all outer loop iterations $k = 1,2,\ldots$. At iteration h, define the event $D(h,\beta)$, termed the set of *β-acceptable solutions*, as

$$\begin{aligned} D(h,\beta) &= \{(\omega(1),\omega(2),\ldots,\omega(h)): \omega(i) \in \Omega, i = 1,2,\ldots,h, f(\omega(i)) \leq \beta \text{ for some } i = 1,2,\ldots,h\} \\ &= \{(\omega(1),\omega(2),\ldots,\omega(h)): \omega(i) \in \Omega, i = 1,2,\ldots,h, \omega(i) \in D_\beta \text{ for some } i = 1,2,\ldots,h\} \end{aligned} \quad (21)$$

where $f(\omega^*) \leq \beta \leq max_{\omega \in \Omega} f(\omega)$, $\omega^* \in G$. Therefore, the complementary event is

$$\begin{aligned} D^c(h,\beta) &= \{(\omega(1),\omega(2),\ldots,\omega(h)): \omega(i) \in \Omega, i = 1,2,\ldots,h, f(\omega(i)) > \beta \text{ for all } i = 1,2,\ldots,h\} \\ &= \{(\omega(1),\omega(2),\ldots,\omega(h)): \omega(i) \in \Omega, i = 1,2,\ldots,h, \omega(i) \notin D_\beta \text{ for all } i = 1,2,\ldots,h\}. \end{aligned} \quad (22)$$

The event $D(h,\beta)$ defines sequences of h solutions that result from the execution of a GHC algorithm over h iterations, where one or more solutions have objective function values less than or equal to β. By definition, $D(h,\beta) \subseteq D(h+1,\beta)$ for all iterations h, hence $\{D(h,\beta)\}$ is a telescoping, non-decreasing sequence of events in h.

From (2) and (3), the *one-step β-acceptable (conditional) probability* is defined as

$$r(j,\beta) = P\{D(j,\beta) \mid D^c(j-1,\beta)\}$$
$$= P\{(\omega(1),\omega(2),\ldots,\omega(j)): \omega(i) \in \Omega, i = 1,2,\ldots,j, f(\omega(i)) > \beta \text{ for all } i = 1,2,\ldots,j-1,$$
$$f(\omega(j)) \le \beta\} / P\{D^c(j-1,\beta)\}. \quad (23)$$

The one-step β-acceptable probability at iteration j provides a finite-time performance measure for the effectiveness of a GHC algorithm, namely the ability of the algorithm to visit an element of the solution space with objective function value less than or equal to β at iteration j given that it has not already visited such a solution over the first j-1 iterations.

The one-step β-acceptable probability can also be used to obtain an expression for the expected number of iterations to visit the set of β-acceptable solutions for the first time. In particular, define the random variable τ_β to represent the number of iterations needed to visit for the first time an element in the set of β-acceptable solutions,

$$\tau_\beta \equiv \min\{j \ge 1 : f(\omega(j)) \le \beta\}. \quad (24)$$

The relationship between τ_β and the one-step β-acceptable probability is described in Lemma 2.

Lemma 2 (Orosz and Jacobson 2002a): Consider a GHC algorithm execution with initial solution generated such that $P\{D^c(0,\beta)\} = 1$. Then

$$P\{\tau_\beta > h\} = \prod_{j=1}^{h} [1 - r(j,\beta)] = P\{D^c(h,\beta)\}. \quad (25)$$

Theorem 18 provides an expression for the conditional expectation of τ_β.

Theorem 18 (Orosz and Jacobson 2002a): Consider a GHC algorithm execution with initial solution generated such that $P\{D^c(0,\beta)\} = 1$. At iteration $h = 1,2,\ldots$, if $P\{\tau_\beta \le h\} > 0$ for some $\beta > f(\omega^*)$, $\omega^* \in G$, then

$$E[\tau_\beta \mid \tau_\beta \le h] = h - \sum_{\tau=1}^{h-1} [1 - \prod_{j=1}^{\tau} [1-r(j,\beta)]] / [1 - \prod_{j=1}^{h} [1-r(j,\beta)]].$$

The primary difficulty with the expression in Theorem 18 is that it is conditional upon the GHC algorithm run length. We now consider the CSA algorithm and show how it is possible to use the expression in Theorem 18 to obtain upper and lower bound estimators for $E[\tau_\beta]$.

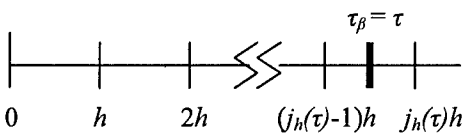
Properties of CSA algorithms are used to obtain upper and lower bounds for $E[\tau_\beta]$. For the purpose of this analysis, and without loss of generality, assume that only outer and middle iterations are considered, and hence, inner iterations are ignored. Therefore, unless otherwise noted, the total iterations refer to only outer and middle iterations. For practical purpose, this means that any expression for the expected number of iterations to reach a β-acceptable solution would need to be rescaled by the number of inner iterations executed, to determine the total number of outer, middle, and inner iteration executed.

To obtain upper and lower bounds for $E[\tau_\beta]$, the number of iterations executed can be decomposed into a set of cycles, each of length h, where the cycle number of iteration τ is denoted by $j_h(\tau) = \lceil \tau/h \rceil$, the *ceiling function* for τ / h (i.e., the smallest integer greater than or equal to τ / h). To obtain the cycle in which the algorithm visits any element in the set of β-acceptable solutions for the first time, define the random variable

$$J_h(\tau_\beta) \equiv \min\{j_h(\tau) \ge 1 : j_h(\tau)h \ge \tau_\beta, \tau = 1,2,\ldots\} = \lceil \tau_\beta / h \rceil. \quad (26)$$

Figure 3 depicts the relationship between $j_h(\tau)$ and τ, where $j_h(\tau) = j$ for all values of τ between $(j-1)h+1$ and jh. Therefore, if the set of β-acceptable solutions is visited for the first time between iterations $(j_h(\tau)-1)h+1$ and $j_h(\tau)h$, then $J_h(\tau_\beta) = j_h(\tau)$.

Figure 3: Relationship between $j_h(\tau)$ and τ_β



Theorem 19 uses the random variable $J_h(\tau_\beta)$ to obtain upper and lower bounds for $E[\tau_\beta]$, the expected number of iterations to visit the set of β-acceptable solutions for the first time.

Theorem 19 (Orosz and Jacobson 2002a): Consider a GHC algorithm execution with initial solution generated such that $P\{D^c(0,\beta)\} = 1$. Then

$$1 + h\,[E[J_h(\tau_\beta)] - 1] \le E[\tau_\beta] \le 1 + h\,[E[J_h(\tau_\beta)]]. \quad (27)$$

Theorem 19 provides upper and lower bounds for $E[\tau_\beta]$ that are functions of h and $E[J_h(\tau_\beta)]$. The practical limitation in applying these bounds is that they contain infinite summations. In particular, $E[J_h(\tau_\beta)] = \sum_{j=0}^{+\infty} [\prod_{i=1}^{jh} [1 - r(i,\beta)]]$ (Orosz and Jacobson 2002a), hence these bounds are not easily computable. This problem is difficult to overcome for GHC algorithms in general. However, since CSA algorithms cycle

through a set of temperature parameters, it may be possible to overcome this obstacle. In particular, if the random variable $J_h(\tau_\beta)$ can be modeled as a geometric distribution with parameter $P\{\tau_\beta \le h\}$, hence $E[J_h(\tau_\beta)] = 1/P\{\tau_\beta \le h\}$, the bounds in Theorem 19 simplify to

$$1 + h\,[P\{\tau_\beta > h\} / P\{\tau_\beta \le h\}] \le E[\tau_\beta] \le 1 + h\,[1 / P\{\tau_\beta \le h\}].$$

From Lemma 2, $P\{\tau_\beta \le h\}$ can be estimated using for a CSA algorithm. Therefore, $E[J_h(\tau_\beta)]$ for such an algorithm can be estimated using information obtained from replicating each algorithm's performance over a single cycle (i.e., over the first h iterations). In particular, the probability that the algorithm visits the set of β-acceptable solutions for the first time in cycle m is

$$P\{J_h(\tau_\beta) = m\} = P\{(m-1)h + 1 \le \tau_\beta \le mh\}. \quad (28)$$

Lemma 3 uses (9) to show that if the one-step β-acceptable probability is cyclical over all iterations (i.e., $r(i,\beta) = r(h+i,\beta)$ for all $i = 1,2,\ldots$, where h is the number of iterations in the temperature cycle), then $J_h(\tau_\beta)$ is a geometric random variable with parameter $P\{\tau_\beta \le h\} = 1 - \prod_{i=1}^{h} [1 - r(i,\beta)]$.

Lemma 3 (Jacobson et al. 2005a): Consider a CSA algorithm (with cycle length h) executed with initial solution generated such that $P\{D^c(0,\beta)\}=1$. If $\prod_{i=(j-1)h+1}^{jh} [1 - r(i,\beta)] = \prod_{i=jh+1}^{(j+1)h} [1 - r(i,\beta)]$ for all $j = 1,2,\ldots$, then $J_h(\tau_\beta)$ is a geometric random variable with parameter $P\{\tau_\beta \le h\} = 1 - \prod_{i=1}^{h} [1 - r(i,\beta)]$.

For CSA algorithms, since the temperature cycles (with cycle length h) are identical, then the probability of visiting a solution with objective function value less than or equal to β should be the same between cycles. Once again, though it may be difficult to formally prove this result, it may however be reasonable to assume that the $r(i,\beta)$ are cyclic with cycle length h, and hence, $\prod_{i=(j-1)h+1}^{jh} [1 - r(i,\beta)] = \prod_{i=jh+1}^{(j+1)h} [1 - r(i,\beta)]$ for all $j = 1,2,\ldots$, and the result in Lemma 3 would apply. Theorem 20 provides upper and lower bounds for $E[\tau_\beta]$, under this assumption. Theorem 21 presents a second set of upper and lower bounds under the same assumption.

Theorem 20 (Jacobson et al. 2005a): Consider a CSA algorithm (with cycle length h) executed with initial solution generated such that $P\{D^c(0,\beta)\} = 1$. If $J_h(\tau_\beta)$ is a geometric random variable with parameter $P\{\tau_\beta \le h\}$, then $1 + h\,[P\{\tau_\beta > h\} / P\{\tau_\beta \le h\}] \le E[\tau_\beta] \le 1 + h / P\{\tau_\beta \le h\}$.

Theorem 21 (Jacobson et al. 2005a): Consider a CSA algorithm (with cycle length h) executed with initial solution generated such that $P\{D^c(0,\beta)\} = 1$. If $J_h(\tau_\beta)$ is a geometric random variable with parameter $P\{\tau_\beta \le h\}$, then

$$E[\tau_\beta \mid \tau_\beta \le h]\, P\{\tau_\beta \le h\} + [1 + h / P\{\tau_\beta \le h\}]\, P\{\tau_\beta > h\}$$
$$\le E[\tau_\beta] \le E[\tau_\beta \mid \tau_\beta \le h]\, P\{\tau_\beta \le h\} + [h + 1 + h / P\{\tau_\beta \le h\}]\, P\{\tau_\beta > h\}.$$

Theorem 22 shows that the bounds in Theorem 21are tighter than those in Theorem 20.

Theorem 22 (Jacobson et al. 2005a): Consider a CSA algorithm (with cycle length h) executed with initial solution generated such that $P\{D^c(0,\beta)\}=1$.. If $J_h(\tau_\beta)$ is a geometric random variable with parameter $P\{\tau_\beta \le h\}$, then

$$E[\tau_\beta \mid \tau_\beta \le h]\, P\{\tau_\beta \le h\} + [h + 1 + h / P\{\tau_\beta \le h\}]\, P\{\tau_\beta > h\} \le 1 + h / P\{\tau_\beta \le h\}$$

and

$$E[\tau_\beta \mid \tau_\beta \le h]\, P\{\tau_\beta \le h\} + [1 + h / P\{\tau_\beta \le h\}]\, P\{\tau_\beta > h\} \ge 1 + h\,[P\{\tau_\beta > h\} / P\{\tau_\beta \le h\}].$$

Computational results with the traveling salesman problem to illustrate how the upper and lower bounds for $E[\tau_\beta]$ in Theorem 21 can be estimated. The traveling salesman optimization problem (TSP) is a well-studied NP-hard discrete optimization problem (Lawler et al. 1985). The diversity of applications for the TSP makes it a frequent choice for testing and evaluating the efficiency and effectiveness of algorithms and heuristics for intractable discrete optimization problems. Traditional TSP applications can be found in numerous domains, including vehicle routing and scheduling problems (Hillier and Lieberman 2001), while more recent applications include manipulation of robotics (Balaguer et al. 2000), cutting of industrial components (Foerster and Wäscher 1998), and circuit board design (Kobayashi et al. 1999).

To formally describe the TSP, several definitions are needed (Lawler et al. 1985). Define a *graph* to be a finite set of *vertices*, some pairs of which are joined by *edges*. A *circuit* in a graph is a set of vertices of the graph, such that it is possible to move between vertices so that all vertices are encountered exactly once, finishing at the start. If a circuit contains all the vertices of the graph, it is called a *Hamiltonian circuit (or tour)*. The usage of the term *cities* is interchangeable with the term *vertices*, and if there exists a direct tour between two cities it is equivalent to the existence of an *edge* between the two cities. Using this terminology, the TSP optimization can be formally stated (Garey and Johnson 1979).

Traveling Salesman (Optimization) Problem (TSP):
Instance: Given a set of n cities $C = \{c_1, c_2, \ldots, c_n\}$ and a distance matrix D that represents the cost of traveling between the cities in the set C.

Question: Find a Hamiltonian circuit $\omega = (c_{j1}, c_{j2}, \ldots, c_{jn})$ that minimizes $f(\omega) = \sum_{i=1}^{n-1} D(c_{ji}, c_{ji+1}) + D(c_{jn}, c_{j1})$.

An instance of a TSP is a discrete optimization problem, where the solution space Ω is the set of all possible Hamiltonian circuits (with each circuit consisting of n cities). The objective function value for each solution $\omega \in \Omega$ is the sum of the distances along the circuit. The optimal objective function value corresponds to the shortest Hamiltonian circuit.

To apply a local search algorithm to a TSP instance, a neighborhood function must be defined. There are numerous neighborhood functions that have been devised for the TSP. the most commonly used TSP neighborhood function is 2-Opt (Croes 1958), or more generally, the λ-Opt neighborhood functions (Lin and Kernighan 1973). A modified version of λ-Opt randomly selects λ unique and nonadjacent cities from the current solution and randomly permutes and reverses the order of these cities such that the new solution is a Hamiltonian circuit. CSA with this modified neighborhood function were used for all the computational experiments reported.

Results are reported with CSA algorithms applied to four TSP instances taken from TSPLIB (Reinelt 1991), ranging in size from 52 cities to 130 cities, with known globally optimal values (see the values under the problem names in Table 1). Each CSA algorithm execution was initialized with a randomly generated solution. The CSA algorithm was applied with two different cycle lengths (h = 1000, 2000), each with N = 100 or 1000 inner loop iterations (depending on which TSP instance was being considered), and cyclical cooling schedule, $T(j+1) = \gamma T(j)$, $j = 1,2,\ldots,h\text{-}1$, which uses a multiplicative cooling factor, γ, similar to cooling schedules that are often used for SA implementations in practice (Henderson et al. 2003), where γ = .99 for h = 1000, and γ = .995 for h = 2000. One thousand replications were executed for each CSA algorithm formulation, with the initial temperature $T(1)$ set as the randomly generated initial solution objective function value.

Since each TSP instance has known globally optimal value, f^*, then the values of β were chosen to be 1%, 2%, 5%, and 10% above this value (i.e., $(1.01)f^*$, $(1.02)f^*$, $(1.05)f^*$, $(1.10)f^*$). Therefore, to estimate the one-step β-acceptable probability at each iteration, for each values of β, the 0-1 indicator function $I_r[f(\omega(i)) \le \beta$ for some $i = 1,2,\ldots,h]$ was obtained for each replication r to determine if a circuit with total distance less than or equal than β was visited over the h iterations (each with N = 100 or 1000 inner loop iterations) of the CSA algorithm. These indicator functions were used to compute the estimator

$$\bar{P}\{\tau_\beta \le h\} = \sum_{r=1}^{1000} I_r[f(\omega'(j)) \le \beta \text{ for some } j = 1,2,\ldots,h] / 1000 \qquad (29)$$

for the four values of β, where $\omega'(j)$ denotes the best solution visited during those N inner loop iterations executed with temperature value $T(j)$. Similarly, for those replications where $f(\omega'(j)) \le \beta$ for some $j = 1,2,\ldots,h$, the actual iteration at which this occurred was used to estimate $E[\tau_\beta \mid \tau_\beta \le h]$. In particular, for replication r, define $j_r(\beta,h) = \min\{j: f(\omega'(j)) \le \beta, j = 1,2,\ldots,h\}$, where $j_r(\beta,h) = 0$ if $f(\omega'(j)) > \beta$ for all $j = 1,2,\ldots,h$. Then

$$\bar{E}[\tau_\beta \mid \tau_\beta \le h] = \sum_{r=1}^{1000} j_r(\beta,h) / \sum_{r=1}^{1000} I_r[f(\omega'(j)) \le \beta \text{ for some } j = 1,2,\ldots,h]. \qquad (30)$$

Lower and upper bound estimates for $E[\tau_\beta]$ using Theorem 21 were then computed by substituting (29) and (30) for $P\{\tau_\beta \le h\}$ and $E[\tau_\beta \mid \tau_\beta \le h]$, respectively.

To compute sample standard deviation estimates for the lower and upper bounds, one hundred independent sets of one thousand independent replications of h iteration cycles (each with N = 100 or 1000 inner loop iterations), each with a randomly generated initial solution, were executed, to obtain a sample of one hundred independent observations for the lower and upper bounds (see Table 7), where the associated point estimates for μ_L and μ_U are denoted by $\bar{L}$ and $\bar{U}$, respectively, and the associated (sample) standard deviation estimates for $\bar{L}$ and $\bar{U}$ are denoted by $s_{\bar{L}}$ and $s_{\bar{U}}$, respectively. All the experiments were executed on a 2.6MHz Pentium IV with 1024MB of RAM. The CPU times (per set of 1000 replications) for the computer experiments for each TSP instance for the lower and upper bound estimation experiments ranged from 353 to 721 CPU seconds for N = 100 and 2993 to 11363 CPU seconds for N = 1000.

To assess the validity of the lower and upper bounds interval estimates reported in Table 7, the CSA algorithms were modified such that several cycles of h iterations (each with N = 100 or 1000 inner loop iterations) were executed until all one thousand replications visited solutions that were within 1% of the globally optimal value, where each cycle was initialized with a randomly generated initial solution. The resulting data was then used to compute point estimates for $E[\tau_\beta]$ and $Var[\tau_\beta]$, denoted by $\bar{E}[\tau_\beta]$ and $s^2[\tau_\beta]$,

respectively; see Table 7. In particular, define c_β to be the number of cycles (each of length h) such that all one thousand replications visited a solution within β of the globally optimal value f^*, for $\beta = (1.01)f^*$, $(1.02)f^*$, $(1.05)f^*$, $(1.10)f^*$. The resulting mean and variance estimators are

$$\bar{E}[\tau_\beta] = \sum_{r=1}^{1000} j_r(\beta, c_\beta h) / 1000 \tag{31}$$

and

$$s^2[\tau_\beta] = \sum_{r=1}^{1000} (j_r(\beta, c_\beta h) - \bar{E}[\tau_\beta])^2 / 999, \tag{32}$$

where $j_r(\beta, c_\beta h) = \min\{j: f(\omega(j)) \le \beta, j = 1,2,\ldots, c_\beta h\}$ for replication $r = 1,2,\ldots,1000$. Therefore, the lower and upper bound point estimates for $E[\tau_\beta]$ are compared to the confidence interval estimates for $E[\tau_\beta]$ obtained using (31) and (32). The CPU times (per set of 1000 replications) for these computer experiments for each TSP instance ranged from 1061 to 7175 CPU seconds for $N = 100$ and 27746 to 1136270 CPU seconds for $N = 1000$.

For simplicity (i.e., not taking into account the standard deviations of the lower and upper bound estimates), the lower bound and upper bounds were defined to provide coverage for $E[\tau_\beta]$ when the 95% confidence interval estimates for $E[\tau_\beta]$ had some overlap in the range between the lower bound and the upper bound points estimates (i.e., $\bar{L} \le \bar{E}[\tau_\beta] + Z_{.025}s[t_\beta]/\sqrt{1000}$ and $\bar{U} \ge \bar{E}[\tau_\beta] - Z_{.025}s[t_\beta]/\sqrt{1000}$, where $Z_{.025}$ is the .025 tail probability value for a standard normal random variable.) When coverage occurred, the point estimates for $E[\tau_\beta]$ in Table 7 are highlighted in bold. From the results in Table 7, for both values of h, thirty of the thirty-two lower and upper bound point estimates provided coverage for $E[\tau_\beta]$. If the values for the sample standard deviation estimates are taken into account, then this fraction of covered values would be slightly higher. These results suggest that the bound estimators provide reasonable estimates for $E[\tau_\beta]$.

Table 7: CSA Algorithm Results for $E[\tau_\beta]$

Problem Instance	β / f^*	$h = 1000$			$h = 2000$		
		$(\bar{L}, s_{\bar{L}})$	$(\bar{U}, s_{\bar{U}})$	$(\bar{E}[\tau_\beta], \frac{s(\tau_\beta)}{\sqrt{1000}})$	$(\bar{L}, s_{\bar{L}})$	$(\bar{U}, s_{\bar{U}})$	$(\bar{E}[\tau_\beta], \frac{s(\tau_\beta)}{\sqrt{1000}})$
Berlin52	1.01	(4543, 37)	(5357, 38)	(**5239**, 149)	(3980, 19)	(5240, 22)	(**4797**, 138)
(7542)	1.02	(3322, 26)	(4079, 27)	(**4097**, 120)	(3284, 13)	(4422, 16)	(**4034**, 109)
N=100	1.05	(1013, 3)	(1358, 4)	(**1322**, 32)	(1606, 2)	(1955, 4)	(**1847**, 31)
	1.10	(700, 0.2)	(716, 0.5)	(**713**, 4)	(1347, 0.2)	(1353, 0.5)	(**1349**, 3)
pr76	1.01	(28447, 546)	(29412, 547)	(**26999**, 892)	(18416, 200)	(20217, 202)	(**19563**, 594
(108159)	1.02	(4439, 36)	(5247, 38)	(**4905**, 147)	(3837, 18)	(5065, 21)	(**4650**, 115)
N=100	1.05	(919, 1)	(1123, 3)	(**1082**, 18)	(1606, 1)	(1732, 2)	(**1712**, 17)
	1.10	(772, 0.1)	(774, 0.2)	(**773**, 1)	(1475, 0.1)	(1475, 0.1)	(**1473**, 1)
kroA100	1.01	(5297, 51)	(6133, 53)	(**6110**, 170)	(7003, 43)	(8536, 46)	(**8589**, 261)
(21282)	1.02	(1607, 6)	(2155, 8)	(**2064**, 52)	(2401, 18)	(3219, 9)	(**3061**, 69)
N=1000	1.05	(833, 0.2)	(878, 1)	(**884**, 8)	(1620, 0.2)	(1641, 1)	(**1635**, 6)
	1.10	(779, 0.1)	(779, 0.1)	(**778**, 1)	(1538, 0.1)	(1538, 0.1)	(**1539**, 1)
ch130	1.01	(371520, 25968)	(372516, 25968)	(304288,10087)	(366587, 21965)	(368574, 21965)	(297724, 9214)
(6110)	1.02	(24096, 394)	(25056, 395)	**(24685, 802)**	(25153, 292)	(27004, 294)	(**27306**, 838)
N=1000	1.05	(1069, 2)	(1384, 3)	**(1363, 26)**	(1839, 1)	(2150, 4)	(**2127**, 31)
	1.10	(828, 0.1)	(829, 0.1)	**(830, 2)**	(1627, 0.1)	(1627, 0.1)	(**1626**, 1)

A procedure is described to estimate lower and upper bounds for the expected number of iterations to visit a β-acceptable solution for the CSA algorithm. Computational results with four traveling salesman problem instances taken from TSPLIB are reported. Work is in progress to develop new estimators for $E[\tau_\beta]$ that can be computed in a single algorithm execution, rather than requiring several algorithm execution replications, as well as provide good estimates for a broader set of cooling schedules. The development of on-line adaptive procedures that can estimate $E[\tau_\beta]$ while the algorithm is being executed would enable such information to be used to determine when the algorithm should be terminated. For example, a single CSA algorithm execution can be decomposed into its cycles, where (29) and (30) can be computed using each cycle as its own replication. Then if an algorithm has been executed for τ iterations, where $\tau > kE[\tau_\beta]$ for some objective function value β and some $k \in Z^+$, and the algorithm has yet to visit a solution with objective function value

below β, the algorithm execution should be terminated or restarted with a new initial solution. Results such as these would be particularly helpful in setting stopping criteria for algorithm runs, since the upper and lower bounds estimates provide reasonable guidelines for assessing when an algorithm execution has gone on "long enough" without having visited a solution with a desired or improved objective function value. In addition, by knowing *a priori* how many iterations it should take to visit a solution with objective function value β, one can then plot these values (upper and lower bound estimates versus β) and fit a regression model to estimate $E[\tau_\beta]$ for values of β with $\overline{P}\{\tau_\beta \leq h\} = 0$. An interesting by-product of such a curve would be a means to estimate the global objective function value of the problem instance, and use this information to guide the execution run length of a CSA algorithm. Extending these results to other hill climbing algorithms would also be extremely useful.

6. Other Research Results

In addition to the results reported above,, several other results were obtained during the period of the grant. These results are briefly discussed here.

In the area of algorithm analysis, Armstrong and Jacobson (2004) introduces semi-data-independent order transformations (SDIOT) such that if problem *A* SDIOT to problem *B* and *B* has a semi-reasonable neighborhood function, where the number of local optima is polynomial, then problem *A* has a semi-reasonable neighborhood function such that the number of local optima is polynomial. A large class of optimization problems is given so that every problem in this class SDIOT to Maximum Clause Weighted Satisfiability (MCWS). Jacobson and Yucesan (2004b) present necessary and sufficient convergence conditions for generalized hill climbing algorithms. These conditions are shown to be equivalent to necessary and sufficient convergence conditions for simulated annealing when the generalized hill climbing algorithm is restricted to simulated annealing. Performance measures are also introduced that permit generalized hill climbing algorithms to be compared using random restart local search. These results identify a solution landscape parameter based on the basins of attraction for local optima that determines whether simulated annealing or random restart local search is more effective in visiting a global optimum. Venkat et al. (2004) introduce a new algorithm that allows decision-makers to take a large group of Pareto optima and obtain a subset of Pareto optima solutions, based on the preferences of a decision-maker. An optimization problem that solves for such quality solutions is formulated. A polynomial Greedy-type heuristic is presented. Computational results are reported that demonstrate the algorithm's performance.

In the area of landscape analysis and the NK Kauffman Model, which has been used in theoretical biology, physics and business organizations to model complex systems with interacting components, Kaul and Jacobson (2006) report global optima results by transforming the problem into a stochastic network model that is closely related to two well-studied problems in operations research. This leads to applicable strategies for explicit computation of bounds on the global optima particularly with K either small or close to N. A general lower bound, which is sharp for K=0, is obtained for the expected value of the global optimum of the NK model. A detailed analysis is provided for the expectation and variance of the global optimum when K=N-1. The lower and upper bounds on the expectation obtained for this case show that there is a wide gap between the values of the local and the global optima. They also indicate that the complexity catastrophe that occurs with the local optima does not arise for the global optima. Kaul and Jacobson (2006) presents new global optima results for the NK model by developing tools for handling dependency in the cases where K grows with N. These results generalize previous work that focused on the analysis of the (independent) case K=N-1. A dependency graph is defined and studied to handle dependencies among underlying random variables in the NK model. Order statistics (with dependencies) and the expected value of the global optima, $E_{N,K}$, are bounded using equitable coloring of the dependency graph. These bounds convert the problem of bounding order statistics of dependent random variables into that of independent random variables while incorporating quantitative information about the mutual dependencies between the underlying random variables. An alternative upper bound on $E_{N,K}$ using direct arguments is also proposed. A detailed analysis of $E_{N,K}$ for K close to N is given for underlying uniform and normal distributions. Finally, for bounded underlying distributions, the global optima are shown to be concentrated around their mean.

Optimal search strategies for conducting reconnaissance, surveillance or search and rescue operations with limited assets are of significant interest to military decision makers. Multiple search platforms with varying capabilities can be deployed individually or simultaneously for these operations (e.g., helicopters, fixed wing or satellite). Due to the timeliness required in these operations, efficient use of available search platforms is critical to the success of such missions. Designing optimal search strategies over multiple search platforms can be modeled and solved as a multiple traveling salesman problem (MTSP). Jacobson et al. (2006b) demonstrate how simultaneous generalized hill climbing algorithms (SGHC) can be used to determine optimal search strategies over multiple search platforms for the MTSP. Computational results with SGHC algorithms applied to the MTSP are reported. These results demonstrate that when limited computing budgets are available, optimal/near-optimal search strategies over multiple search platforms can be obtained more

efficiently using SGHC algorithms compared to other generalized hill climbing algorithms. Applications and extensions of this research to other military applications are also discussed. McLay and Jacobson (2007) study the Integer Knapsack Problem with Set-up Weights (IKPSW), a generalization of the classical Integer Knapsack Problem (IKP), where each item type has a set-up weight that is added to the knapsack if any copies of the item type are in the knapsack solution. The *k*-item IKPSW (*k*IKPSW) is also considered, where a cardinality constraint imposes a value *k* on the total number of items in the knapsack solution. IKPSW and *k*IKPSW have applications in the area of aviation security. This paper provides dynamic programming algorithms for each problem that produce optimal solutions in pseudo-polynomial time. Moreover, four heuristics are presented that provide approximate solutions to IKPSW and *k*IKPSW. For each problem, a Greedy heuristic is presented that produces solutions within a factor of 1/2 of the optimal solution value, and a fully polynomial time approximation scheme (FPTAS) is presented that produces solutions within a factor of ε of the optimal solution value. Time and space requirements for the FPTAS for IKPSW and *k*IKPSW are also reported.

In the area of aviation security system design and analysis, Candalino et al. (2004) introduce a comprehensive cost function that not only includes direct costs associated with the purchase and operation of baggage screening security devices, but also includes indirect costs associated with device errors. A methodology is presented to determine the best selection of baggage screening security devices that minimizes the expected annual total cost of a baggage screening strategy. Computational experiments with this methodology are presented. Jacobson et al. (2005b) analyze checked baggage screening strategies that incorporate the effects of deterrence for explosive detection systems (EDSs) deployed at airports. Cost models for these strategies are presented that incorporate the cost of purchasing, operating, and maintaining an EDS, the number of checked bags available to be screened, and the numbers of selectee and non-selectee checked bags actually screened over a one-year period. The model also includes the effect of deterrence on the level of threat at an airport. The cost models provide a quantitative tool to assess the strategy of 100% screening of all checked bags, as set forth by the United States Aviation and Transportation Security Act. The key conclusion from this analysis is that the cost effectiveness of 100% screening of all checked bags compared to screening only checked selectee bags depends on how quickly an increase in the number of checked bags screened reduces the threat level (i.e., the deterrence effect). Comparing the expected direct cost per expected prevented attack to the expected cost of an aviation terrorist incident provides an indication of the cost effectiveness of 100% checked bag screening. Jacobson et al. (2005c) present NP-complete decision problems concerning the deployment and utilization of baggage screening security devices. These problems incorporate three different deployment performance measures: uncovered baggage segments, uncovered flight segments, and uncovered passenger segments. Integer programming models are formulated to address optimization versions of these problems and to identify optimal baggage screening security device deployments (i.e., determine the number and type of baggage screening security devices that should be placed at different airports, and determining which baggage should be screened with such devices). The models are illustrated with an example that incorporates data extracted from the Official Airline Guide (OAG). Jacobson et al. (2005d) report results of using operations research methodologies to design and analyze aviation security baggage screening systems. In the aftermath of the tragic events of September 11, 2001, numerous changes have been made to aviation security policy and operations throughout the nation's airports. The allocation and utilization of checked baggage screening devices is a critical component in aviation security systems. This paper formulates problems that model multiple sets of flights originating from multiple stations (e.g., airports, terminals), where the objective is to optimize a baggage screening performance measure subject to a finite amount of resources. These measures include uncovered flight segments (UFS) and uncovered passenger segments (UPS). Three types of multiple station security problems are identified and their computational complexity is established. The problems are illustrated on two examples that use data extracted from the Official Airline Guide. The examples indicate that the problems can provide widely varying solutions based on the type of performance measure used and the restrictions imposed by the security device allocations. Moreover, the examples suggest that the allocations based on the UFS measure also provide reasonable solution with respect to the UPS measure; however, the reverse may not be the case. This suggests that the UFS measure may provide more robust screening device allocations. Jacobson et al. (2006a) evaluate the cost-effectiveness of the explosive detection technologies currently deployed to screen checked baggage as well as new technologies that could be used in the future. Both single-device and two-device systems are considered. In particular, the expected annual direct cost of using these devices for 100% checked baggage screening under various scenarios is obtained and the tradeoffs between using single-device and two-device strategies are studied. The expected number of successful threats under the different checked baggage screening scenarios with 100% checked baggage screening is also obtained. Lastly, a risk-based screening strategy proposed in the literature is analyzed. The results reported suggest that for the existing security setup, with current device costs and probability parameters, single-device systems are less costly and

have fewer expected number of successful threats than two-device systems due to the way the second device affects the alarm or clear decision. The risk-based approach is found to have the potential to significantly improve security. The cost model introduced provides an effective tool for the execution of cost-benefit analyses of alternative device configurations for aviation checked baggage security screening. McLay et al. (2006) introduce the Multilevel Allocation Problem (MAP), which models the screening of passengers and baggage in a multilevel aviation security system. A passenger is screened by one of several classes, each of which corresponds to a set of procedures using security screening devices, where passengers are differentiated by their perceived risk levels. Each class is defined in terms of its fixed cost (the overhead costs), its marginal cost (the additional cost to screen a passenger), and its security level. The objective of MAP is to assign each passenger to a class such that the total security is maximized subject to passenger assignments and budget constraints. This paper shows that MAP is NP-hard and introduces a Greedy heuristic that obtains approximate solutions to MAP that use no more than two classes. Examples are constructed using data extracted from the Official Airline Guide. Analysis of the examples suggests that fewer security classes for passenger screening may be more effective and that using passenger risk information can lead to more effective security screening strategies. McLay et al. (2007) introduce the Multilevel Passenger Screening Problem (MPSP). In MPSP, a set of classes are available for screening passengers, each of which corresponds to several device types for passenger screening, where each device type has an associated capacity and passengers are differentiated by their perceived risk levels. The objective of MPSP is to use prescreening information to determine the passenger assignments that maximize the total security subject to capacity and assignment constraints. MPSP is illustrated with examples that incorporate flight schedule and passenger volume data extracted from the Official Airline Guide.

In the area of random number generation, Smith and Jacobson (2005) introduce and study a discrete optimization problem related to the alias method for discrete random number generation. The alias method is an efficient method to generate several random variates from a discrete probability distribution. The efficiency of the alias method can be improved by designing the alias table such that the expected number of computations that must be performed per value generated is minimized. The problem of optimizing the construction of the alias table is proven to be strongly NP-Hard, even if either of two variations of the alias method relaxing the alias table generation restrictions are used. Integer programming formulations describing these three optimization problems are presented, and insights regarding necessary optimality criteria and relationships among their optimal solutions are discussed.

REFERENCES

E. Aarts, J. Korst, 2002, "Selected Topics in Simulated Annealing," Chapter 1 in *Essays and Surveys on Metaheuristics* (P. Hansen, C.C. Ribeiro, Editors), Kluwer Academic Publishers, Norwell, Massachusetts, 1-37.

E. Aarts, L.K. Lenstra. 1997. *Local Search in Combinatorial Optimization*, Wiley and Sons, New York.

N. Abboud, M. Sakawa, M. Inuiguchi, 1998, "School Scheduling Using Threshold Accepting," *Cybernetics and Systems*, 29(6): 593-611.

R.K. Ahuja, O. Ergun, J.B. Orlin, A.P. Punnen, 2003, "A Survey of Very Large-Scale Neighborhood Search Techniques," *Discrete Applied Mathematics*, 23, 75 – 102.

S. Anily, A. Federgruen, 1987, "Simulated Annealing Methods with General Acceptance Probabilities," *Journal of Applied Probability*, 24, 657-667.

D.E. Armstrong, S.H. Jacobson, 2004, "Polynomial Transformations and Data-Independent Neighborhood Functions," *Discrete Applied Mathematics*, 143(1-3), 272-284.

D.E. Armstrong, S.H. Jacobson, 2005, "Data Independent Neighborhood Functions and Strict Local Optima," *Discrete Applied Mathematics*, 146(3), 233-243.

D.E. Armstrong, S.H. Jacobson, 2006a, "Order Preserving Reductions and Polynomial Improving Paths," *Operations Research Letters* 34(1), 9-16.

D.E. Armstrong, S.H. Jacobson, 2006b, "Analyzing the Complexity of Finding Good Neighborhood Functions for Local Search Algorithms," *Journal of Global Optimization*, 36(2), 219-236.

G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, 1999, "Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties," Springer-Verlag, Berlin.

G. Ausiello, P. Crescenzi, M. Protasi, 1995, "Approximate Solution of NP Optimization Problems," *Theoretical Computer Science*, 150, 1-55.

C. Balaguer, A. Gimenez, J.M., Pastor, V.M. Padrón and M. Abderrahim, 2000, "A Climbing Autonomous Robot for Inspection Applications in 3D Complex Environments," *Robotica*, 18, 287-297.

J.W. Barnes, J.B. Chambers, 1995, "Solving the Job-Shop Scheduling Problem with Tabu Search," *IIE Transactions*, 27, 257-263.

P. Billingsley, 1979, *Probability and Measure*, John Wiley and Sons, New York.

T.J. Candalino, J.E. Kobza, S.H. Jacobson, 2004, "Designing Optimal Aviation Baggage Screening Systems using Simulated Annealing," *Computers and Operations Research*, 31(10), 1753-1767.

O. Cantoni, R. Cerf, 1997, "The Exit Path of a Markov Chain with Rare Transitions," *ESAIM: Probability and Statistics,* 1, 95-144.

I. Charon, O. Hudry, 2001, "The Noising Method: A Generalization of Some Metaheuristics," *European Journal of Operational Research*, 135(1), 86-101.

T.S. Chiang, Y.Y. Chow, 1989, "A Limit-Theorem for a Class of Inhomogeneous Markov-Processes," *Annals of Probability,* 17, 1483-1502.

H. Cohn, M. Fielding, 1999, "Simulated Annealing: Searching for an Optimal Temperature Schedule," *SIAM Journal of Optimization*, 9(3), 779-802.

S.A. Cook, 1971, "The Complexity of Theorem-Proving Procedures," *Proceedings of the Third Annual ACM Symposium on Theory of Computing, Association for Computing Machinery*, New York, 151-158.

P. Crescenzi, L. Trevisan, 2000, "On Approximation Scheme Preserving Reducibility and its Applications," *Theory of Computing Systems*, 33, 1-16.

G.A. Croes, 1958, "A Method for Solving Transportation-Salesman Problems," *Operations Research*, 6, 791-812.

M.P. Desai, 1999, "Some Results Characterizing the Finite Time Behaviour of the Simulated Annealing Algorithm," *Sadhana*, 24, 317-337.

G. Dueck, T. Scheuer, 1990, "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing," *Journal of Computational Physics,* 90, 161-175.

U. Faigle, W. Kern, 1992, "Some Convergence Results for Probabilistic Tabu Search," *ORSA Journal on Computing*, 4(1), 32-37.

U. Faigle, R. Schrader, 1988, "On the Convergence of Stationary Distributions in Simulated Annealing algorithms," *Information Processing Letters,* 27, 189-194.

A.G. Ferreira, J. Zerovnik, 1993, "Bounding the Probability of Success of Stochastic Methods for Global Optimization," *Computers with Mathematics Applications*, 25(10/11), 1-8.

M. Fielding, 2000, "Simulated Annealing with an Optimal Fixed Temperature," *SIAM Journal of Optimization*, 11(2), 289-307.

M. Fleischer, S.H. Jacobson, 1999, "Information Theory and the Finite-Time Behavior of the Simulated Annealing Algorithm: Experimental Results," *INFORMS Journal on Computing,* 11, 35-43.

H. Foerster and G. Wascher, 1998, "Simulated Annealing for Order Spread Minimization in Sequencing Cutting Patterns," *European Journal of Operational Research,* 126, 106-130.

B.L. Fox, 1993, "Integrating and Accelerating Tabu Search, Simulated Annealing, and Genetic Algorithms," *Annals of Operations Research*, 41, 47-67.

A. Franz K.H. Hoffmann, P. Salamon, 2001, "Best Possible Strategy for Finding Ground States," *Physical Review Letters*, 86(23), 5219-5222.

M.R. Garey, D.S. Johnson, 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York.

F. Glover, 1977, "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, 8, 156-166.

F. Glover, 1986, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research*, 13(5), 533-549.

F. Glover, 1989, "Tabu Search-Part I," *ORSA Journal on Computing*, 1(3), 190-206.

F. Glover, M. Laguna. 1997. *Tabu Search*. Kluwer Academic Publishing, Norwell, MA.

J. Gu, 1997, "Multispace Search for Satisfiability and NP-hard Problems," D. Du, J. Gu, P.M. Pardalos, Eds. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 35: Satisfiability Problem: Theory and Applications: Proceedings of a DIMACS Workshop, March 11-13, 1996.* American Mathematical Society. Providence, RI. 407-517.

B. Hajek, 1988, "Cooling Schedules for Optimal Annealing," *Mathematics of Operations Research,* 13, 311-329.

D. Henderson, S.H. Jacobson, A.W. Johnson, 2003, "The Theory and Practice of Simulated Annealing," F. Glover, G. Kochenberger, Eds. *State-of-the-Art Handbook in Metaheuristics*. Kluwer Academic Publishing, Norwell, MA. 287-319.

F.S. Hillier, G.J. Lieberman, 2001, *Introduction to Operations Research*, McGraw Hill, Boston, Massachusetts.

D. Isaacson, R. Madsen, 1985, *Markov Chains Theory and Applications*. Robert E. Krieger Publishing Co., Inc., Malabar, FL.

S.H. Jacobson, S.N. Hall, L.A. McLay, J.E. Orosz, 2005a, "Performance Analysis of Cyclic Simulated Annealing Algorithms," *Methodology and Computing in Applied Probability*, 7(2), 183-201.

S. H. Jacobson, T. Karnani, J.E. Kobza, 2005b, "Assessing the Impact of Deterrence on Aviation Checked

Baggage Screening Strategies," *International Journal of Risk Assessment and Management*, 5(1), 1-15.

S.H. Jacobson, T. Karnani, J.E. Kobza, L. Ritchie, 2006a, "A Cost-Benefit Analysis of Alternative Device Configurations for Aviation Checked Baggage Security Screening," *Risk Analysis* 26(2), 297-310.

S.H. Jacobson, L.A. McLay, S.N. Hall, D. Henderson, D.E. Vaughan, 2006b, "Optimal Search Strategies Using Simultaneous Generalized Hill Climbing Algorithms," *Mathematical and Computer Modelling*, 43(9-10), 1061-1073.

S.H. Jacobson, L.A. McLay, J.E. Kobza, J.M. Bowman, 2005c, "Modeling and Analyzing Multiple Station Baggage Screening Security System Performance," *Naval Research Logistics*, 52(1), 30-45.

S.H. Jacobson, L.A. McLay, J.L. Virta, J.E. Kobza, 2005d, "Integer Program Models for the Deployment of Airport Baggage Screening Security Devices," *Optimization and Engineering* 6(3), 339-359.

S.H. Jacobson, D. Solow, 1993, "The Effectiveness of Finite Improvement Algorithms for Finding Global Optima," *Zeitschrift fur Operations Research (ZOR) -- Methods and Models of Operations Research*, 37(3), 257-272.

S.H. Jacobson, K.A. Sullivan, A.W. Johnson, 1998, "Discrete Manufacturing Process Design Optimization using Computer Simulation and Generalized Hill Climbing Algorithms," *Engineering Optimization*, 31, 247-260.

S.H. Jacobson, E. Yucesan, 2004a, "Global Optimization Performance Measures for Generalized Hill Climbing Algorithms," *Journal of Global Optimization*, 29, 177-193.

S.H. Jacobson, E. Yucesan, 2004b, "Analyzing the Performance of Generalized Hill Climbing Algorithms," *Journal of Heuristics*, 10, 387-405.

A.W. Johnson, S.H. Jacobson, 2002a, "On the Convergence of Generalized Hill Climbing Algorithms," *Discrete Applied Mathematics,* 119(1-2), 37-57.

A.W. Johnson, S.H. Jacobson, 2002b, "A Class of Convergent Generalized Hill Climbing Algorithms," *Applied Mathematics and Computation*, 125(2-3), 359-373.

D.S. Johnson, C.H. Papadimitriou, M. Yannakakis, 1988, "How Easy is Local Search?" *Journal of Computers and Systems Science*, 37(1), 79-100.

H. Kaul, S.H. Jacobson, 2006, "Global Optima Results for the Kauffman NK Model," *Mathematical Programming*, 106(2), 319-338.

Kaul, H., Jacobson, S.H., 2007, "New Global Optima Results for the Kauffman NK Model: Handling Dependency," *Mathematical Programming*, 108(2-3), 475-494.

S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi, 1983, "Optimization by Simulated Annealing," *Science,* 220, 671-680.

S. Kobayashi, M. Edahiro, M. Kubo, 1999, "A VLSI Scan-Chain Optimization Algorithm for Multiple Scan-Paths," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Science*, 11, 2499-2504.

E.L. Lawler, L.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, 1985, *The Traveling Salesman Problem*. John Wiley and Sons, Chichester, U.K.

S. Lin, B.W. Kernighan, 1973, "An Effective Heuristic for the Traveling Salesman Problem," *Operations Research,* 21, 498-516.

R. Marti, 2003, "Multi-start Methods," Chapter 12 in *Handbook on Metaheuristics*, F. Glover and G. Kochenberger, Editors, Kluwer Academic Publishers, Norwell, Massachusetts, 355-368.

C. Mazza, 1992, "Parallel Simulated Annealing," *Random Structures and Algorithms,* 3, 139-148.

L.A. Mclay, S.H. Jacobson, 2007, "Integer Knapsack Problems with Set-Up Weights," *Computational Optimization and Applications* (Accepted).

L.A. McLay, S.H. Jacobson, J.E. Kobza, 2006, "A Multilevel Passenger Prescreening Problem for Aviation Security," *Naval Research Logistics*, 53(3), 183-197.

L.A. McLay, S.H. Jacobson, J.E. Kobza, 2007, "Integer Programming Models and Analysis for a Multilevel Passenger Screening Problem, "*IIE Transactions*, 39(1), 73-81.

D. Mitra, F. Romeo, A.L. Sangiovanni-Vincentelli, 1986, "Convergence and Finite-time Behavior of Simulated Annealing," *Advances in Applied Probability,* 18, 747-771.

V. Nissen, H. Paul, 1995, "A Modification of Threshold Accepting and its Application to the Quadratic Assignment Problems," *OR Spekrum*, 17(2-3), 205-210.

A. Nolte, R. Schrader, 2001, "A Note on the Finite Time Behavior of Simulated Annealing," *Mathematics of Operations Research,* 25(3), 476-484.

J.E. Orosz, S.H. Jacobson, 2002a, "Finite-time Performance Analysis of Static Simulated Annealing Algorithms," *Computational Optimization and Applications*, 21(1), 21-53.

Orosz, J.E., S.H. Jacobson, 2002b, "Analysis of Static Simulated Annealing Algorithms," *Journal of Optimization Theory and Application*, 115(1), 165-182.

C.H. Papadimitriou, K. Steilglitz, 1978, "Some Examples of Difficult Traveling Salesman Problems," *Operations Research*, 26, 434-44.

C.H. Papadimitriou, K. Steilglitz, 1982, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ.

G. Reinelt, 1991, "A Traveling Salesman Problem Library", *ORSA Journal on Computing*, 3(4), 376-385

V. Rodl, C.A. Tovey, 1987, "Multiple Optima in Local Search," *Journal of Algorithms*, 8, 250-259.

H.L. Royden, 1988. *Real Analysis*. Macmillan Publishing Company, New York.

S. Savage, P. Weiner, A. Bagchi, 1976, "Neighborhood Search Algorithms for Guaranteeing Optimal Traveling Salesman Tours Must be Inefficient," *Journal of Computer Systems Science*, 12, 25-35.

B. Selman, H. Levesque, D. Mitchell, 1992, A New Method for Solving Hard Satisfiability Problems," *Proceedings of the Tenth National Conference on Artificial Intelligence*. AAAI Press. San Jose, CA. 440-446.

J.C. Smith, S.H. Jacobson, 2005, "An Analysis of the Alias Method for Discrete Random Number Generation, " *INFORMS Journal on Computing*, 17(3), 321-327..

S. Smith, C.C. Cheng, 1993, "Slack-based Heuristics for Constraint Satisfaction Scheduling," *Proceedings of the Eleventh National Conference on Artificial Intelligence*. AIII Press. Washington, DC. 440-446.

R. Srichander, 1995, "Efficient Schedules for Simulated Annealing," *Engineering Optimization,* 24, 161-176.

K.A. Sullivan, S.H. Jacobson, 2001, "A Convergence Analysis of Generalized Hill Climbing Algorithms" *IEEE Transactions on Automatic Control*, 46(8), 1288-1293.

C.A. Tovey, 1985, "Hill Climbing with Multiple Local Optima," *SIAM Journal of Algebraic Discrete Methods*, 6, 384-393.

D.E. Vaughan, S.H. Jacobson, D.E. Armstrong, 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization using Generalized Hill Climbing Algorithms, *ASME Journal of Mechanical Design*, 122, 164-171.

D.E. Vaughan, S.H. Jacobson, 2004, "Formulating the Meta-heuristic Tabu Search in the Generalized Hill Climbing Framework," *Methodology and Computing in Applied Probability*, 6, 343-354.

D.E. Vaughan, S.H. Jacobson, S.N. Hall, L.A. McLay, 2005, "Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problems," *INFORMS Journal on Computing*, 17(4), 438-450.

D.E. Vaughan, S.H. Jacobson, H Kaul, 2007, "Analyzing the Performance of Simultaneous Generalized Hill Climbing Algorithms," *Computational Optimization and Applications* (Accepted).

V. Venkat, S.H. Jacobson, J.A. Stori, 2004, "A Post-Optimality Analysis Algorithm for Multi-Objective Optimization," *Computational Optimization and Applications*, 28(3), 357-372.

V.G. Vizing, 1977, "Complexity of the Traveling Salesman Problem in the Class of Monotonic Improvement Algorithms, *Cybernetics*, 13, 623-626.

M. Yannakakis, 1997, "Computational Complexity," in: J.K. Lenstra and E. Aarts, eds., Local Search in Combinatorial Optimization (John Wiley & Sons, Chichester, 1997) 19-55.

X. Yao, 1995, "A New Simulated Annealing Algorithm," *International Journal of Computer Mathematics*, 56, 161-168.

X. Yao, G. Li, 1991, "General Simulated Annealing," *Journal of Computer Science and Technology*, 6, 329-338.

W. Zhang, T.G. Dietterich, 1997, "Solving Combinatorial Optimization Tasks by Reinforcement Learning: A General Methodology Applied to Resource-constrained Scheduling," Technical Report, Department of Computer Science, Oregon State University, Corvallis, OR.

CONTRIBUTING PERSONNEL

The principal investigator for this project, Sheldon H. Jacobson, Ph.D., has devoted both academic year time and summer time in each of the three years of this project. Colonel Darrall Henderson, US Army, Ph.D., Derek E. Armstrong, Ph.D., Los Alamos National Laboratory, and Diane E. Vaughan, Ph.D., Los Alamos National Laboratory, have provided input on this project. Laura A. McLay, Ph.D., Hemanshu Kaul, Ph.D., and Major Shane N. Hall, USAF, PhD., were Ph.D. students of the principal investigator, who worked on aspects of this project. Mr. Alex Nikolaev, Mr. Gio Kao, and Mr. Adrian Lee are current graduate students of the principal investigator who have also contributed to various aspects of this project.

TRANSITIONS

Extensive interactions with Austral Engineering and Software, Incorporated, have resulted in generalized hill climbing algorithm software code and post-optimality analysis software code (for multi-critieria optimization) being shared with them. These algorithms and software are available for them to use in support of their SBIR activities with the military (Contact: Daniel Allwine).

Research on deterrence modeling and analysis has been transitioned to the Homeland Security Institute within the United States Department of Homeland Security (Contact: Marc Thibault, Ph.D.).

PUBLICATIONS

The following is a list of published papers in refereed journals or book chapters that are related to the research effort supported in whole or in part by this grant.

Armstrong, D.E., Jacobson, S.H., 2004, "Polynomial Transformations and Data Independent Neighborhood Functions," *Discrete Applied Mathematics*, 143(1-3), 272-284.

Armstrong, D.E., Jacobson, S.H., 2005, "Data Independent Neighborhood Functions and Strict Local Optima," *Discrete Applied Mathematics*, 146(3), 233-243

Armstrong, D.E., Jacobson, S.H., 2006, "Order Preserving Reductions and Polynomial Improving Paths," *Operations Research Letters*, 34(1), 9-16.

Armstrong, D.E., Jacobson, S.H., 2006, "Analyzing the Complexity of Finding Good Neighborhood Functions for Local Search Algorithms," *Journal of Global Optimization*, 36(2), 219-236.

Candalino, T.J., Kobza, J.E., Jacobson, S.H., 2004, "Designing Optimal Aviation Baggage Screening Systems using Simulated Annealing," *Computers and Operations Research*, 31(10), 1753-1767.

Jacobson, S.H., Hall, S.N., McLay, L.A., Orosz, J.E., 2005, "Performance Analysis of Cyclic Simulated Annealing Algorithms," *Methodology and Computing in Applied Probability*, 7(2), 183-201.

Jacobson, S.H., Karnani, T., Kobza, J.E., 2005, "Assessing the Impact of Deterrence on Aviation Checked Baggage Screening Strategies," *International Journal of Risk Assessment and Management*, 5(1), 1-15.

Jacobson, S.H., Karnani, T., Kobza, J.E., Ritchie, L., 2006, "A Cost-Benefit Analysis of Alternative Device Configurations for Aviation Checked Baggage Security Screening," *Risk Analysis*, 26(2), 297-310.

Jacobson, S.H., McLay, L.A., Virta, J.L., Kobza, J.E., 2005, "Integer Program Models for the Deployment of Airport Baggage Screening Security Devices," *Optimization and* Engineering, 6(3), 339-359.

Jacobson, S.H., McLay, L.A., Hall, S.N., Henderson, D., Vaughan, D.E., 2006, "Optimal Search Strategies Using Simultaneous Generalized Hill Climbing Algorithms," *Mathematical and Computer Modelling*, 43(9-10), 1061-1073.

Jacobson, S.H., Yucesan, E., 2004, "Global Optimization Performance Measures for Generalized Hill Climbing Algorithms," *Journal of Global Optimization*, 29(2), 177-193.

Jacobson, S.H., Yucesan, E., 2004, "Analyzing the Performance of Generalized Hill Climbing Algorithms," *Journal of Heuristics*, 10(4), 387-405.

Jacobson, S.H., McLay, L.A., Kobza, J.E., Bowman, J.M., 2005, "Modeling and Analyzing Multiple Station Baggage Screening Security System Performance," *Naval Research Logistics*, 52(1), 30-45.

Kaul, H., Jacobson, S.H., 2006, "Global Optima Results for the Kauffman NK Model," *Mathematical Programming*, 106(2), 319-338.

Kaul, H., Jacobson, S.H., 2007, "New Global Optima Results for the Kauffman NK Model: Handling Dependency," *Mathematical Programming*, 108(2-3), 475-494.

Mclay, L.A., Jacobson, S.H., 2007, "Integer Knapsack Problems with Set-Up Weights," *Computational Optimization and Applications* (Accepted).

McLay, L.A., Jacobson, S.H., Kobza, J.E., 2006, "A Multilevel Passenger Prescreening Problem for Aviation Security," *Naval Research Logistics*, 53(3), 183-197.

McLay, L.A., Jacobson, S.H., Kobza, J.E., 2007, "Integer Programming Models and Analysis for a Multilevel Passenger Screening Problem," *IIE Transactions*, 39(1), 73-81.

Smith, J.C., Jacobson, S.H., 2005, "An Analysis of the Alias Method for Discrete Random Number Generation," *INFORMS Journal on Computing*, 7(3), 321-327.

Vaughan, D.E., Jacobson, S.H., 2004, "Formulating the Meta-Heuristic Tabu Search in the Generalized Hill Climbing Framework," *Methodology and Computing in Applied* Probability, 6(3), 343-354.

Vaughan, D.E., Jacobson, S.H., Hall, S.N., McLay, L.A., 2005, "Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problems," *INFORMS Journal on Computing*, 7(4), 438-450.

Vaughan, D.E., Jacobson, S.H., Kaul, H., 2007, "Analyzing the Performance of Simultaneous Generalized Hill Climbing Algorithms," *Computational Optimization and Applications,* (Accepted).

Venkat, V., Jacobson, S.H., Stori, J.A., 2004, "A Post-Optimality Analysis Algorithm for Multi-Objective Optimization," *Computational Optimization and Applications*, 28(3), 357-372.

HONORS/AWARDS

Sheldon H. Jacobson, Ph.D., was renamed a *Willett Faculty Scholar in the College of Engineering* at the University of Illinois for the 2005-2008 academic years. This designation is given to tenured faculty who are excelling in their contributions to the university. Approximately twenty faculty have been awarded this distinction, based on their record of research accomplishments and achievements.

Laura A. McLay, Ph.D., was awarded honorable mention in the 2006 INFORMS Computing Society Student Paper Competition, based on research reported in her paper, "Integer Knapsack Problems with Set-Up Weights."

Laura A. McLay, Ph.D., was awarded the 2005 Harper Safety Award from the Department of Mechanical and Industrial Engineering at the University of Illinois.

Major Shane N. Hall, USAF, Ph.D. was awarded the 2006 Harper Safety Award from the Department of Mechanical and Industrial Engineering at the University of Illinois.